Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DE COMPUTADORES

Localización mellorada de recursos a partir de balizas e rastrexadores Bluetooth Low Energy

Estudante: Sergio Villodas Zapata
Dirección: Carlos Vázquez Regueiro

A Coruña, febreiro de 2021.

Dedicado a toda a xente que fixo posible este proxecto.

Agradecementos

Grazas a todos os profesores que tiven e que me axudaron a aprender, a Carlos por axudarme con este traballo e pola súa dedicación, á miña familia polo seu apoio incondicional e aos meus compañeiros e amigos por todos os momentos vividos durante este bonito paso pola FIC.

Moitas grazas de corazón.

Resumo

Neste proxecto desenvolveuse un sistema de localización, en espazos interiores, de dispositivos Bluetooth Low Energy (BLE). O sistema baséase en varios rastrexadores de baixo custo (Raspberry) que se encargan de escanear os dispositivos BLE e de enviar os datos dos escaneos a un servidor de aplicacións mediante MQTT. O servidor procesa os datos dos escaneos e calcula en tempo real a posición de cada baliza BLE a partir da distancia a cada rastrexador, inferida a través da potencia medida (o RSSI). Si hai datos de suficientes rastrexadores emprégase a multilateración, se non o radio de distancia.

No sistema almacénanse as balizas (dispositivos BLE) que queremos detectar, a información sobre os rastrexadores que imos usar (sobre todo a súa localización no entorno), os datos de todos os escaneos (de cada rastrexador sobre cada baliza coa frecuencia estipulada) e tamén a propia configuración do sistema (o método de resolución da multilateración, o factor ambiental, o tamaño do intervalo de integración, etc.). Estes datos pódense consultar cunha API REST e sobre todo mediante unha interface web coa que se pode visualizar, en tempo real, as balizas e a súa traxectoria calculada cada certo intervalo de tempo.

Para validar o noso sistema, realizáronse probas de conexión da rede e da independencia de todo o sistema, unha calibración dos dispositivos empregados, e por último, probas de localización das baliza no entorno. Demostramos que o sistema proposto é capaz de localizar balizas BLE en interiores, tanto si están en posición estática como en movemento, empregando únicamente hardware de baixo custo, cunha arquitectura modular, escalable, independente de calquer infraestrutura, e cun erro nas posicións similar ao estado da técnica actual.

Abstract

In this project, a system for locating Bluetooth Low Energy (BLE) devices in indoor spaces was developed. The system is based on several low-cost (Raspberry) trackers that are responsible for scanning BLE devices and sending the scan data to an application server using MQTT. The server processes the scan data and calculates in real time the position of each BLE beacon from the distance to each tracker, inferred through the measured power (the RSSI). If there is data from sufficient trackers multilateration is used, if not the radius of distance.

The system stores the beacons (BLE devices) that we want to detect, the information about the trackers that we will use (especially their location in the environment), the data of all scans (of each tracker on each beacon with the stipulated frequency) and also the system configuration itself (the multilateral resolution method, the environmental factor, the size of the integration interval, etc.). This data can be consulted with a API REST and above

all through a web interface with which you can view, in real time, the beacons and their calculated trajectory every certain time interval.

To validate our system, network connection and system-wide independence tests, a calibration of the devices used, and finally, beacon location tests in the environment were performed. We show that the proposed system is capable of locating BLE beacons indoors, whether static or moving, using only low-cost hardware, with a modular, scalable architecture, independent of any infrastructure, and with a state-like position error. of the current technique.

Palabras chave:

- Bluetooth de baixa enerxía
- BLE
- Raspberry Pi
- Localización en interiores
- MQTT
- Rede en malla

Keywords:

- Bluetooth Low Energy
- BLE
- Raspberry Pi
- Indoor positioning system
- MQTT
- Mesh network

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

Neste capítulo exponse as liñas mestras do traballo, os obxectivos e o contexto. Tamén preséntase a proposta do sistema de localización e a estrutura da memoria.

1.1 Contexto

Actualmente os sistemas de localización pouco nos sorprenden, que Google Maps nos localice en calquera lugar non é estraño. Estamos habituados ás tecnoloxías de posicionamento en exteriores e agora case non saberíamos vivir sen elas. Sen dúbida, o GPS e outras tecnoloxías por satélite semellantes revolucionaron o mundo.

Pero, que ocorre co posicionamento en lugares interiores? Os sistemas de localización por satélite non funcionan nestes lugares, pero existen outras tecnoloxías que si o fan, aínda que é necesario usar certa infraestrutura. Falamos dos dispositivos Bluetooth Low Energy (BLE), uns dispositivos que funcionan con baterías de larga duración e que se convertiron no pilar do posicionamento en interiores.

A tecnoloxía Bluetooth Low Energy foi un importante avance na creación de aplicacións co estándar Bluetooth, xa que permite un menor consumo de enerxía, un mellor alcance e máis ancho de banda. O que fai que BLE sexa tan interesante fronte a outros estándares é a súa sinxeleza para implementar a comunicación entre pequenos dispositivos e unha aplicación [1]. Por outra parte, unha tecnoloxía que podería usarse en aplicacións semellantes sería RFID, pero BLE ten a vantaxe de que está dispoñible inmediatamente sobre os dispositivos móbiles sen necesidade de hardware, é máis barato e o seu alcance e maior. [2]

Toda esta tecnoloxía da que estamos a falar permítenos realizar aplicacións con sistemas de localización en interiores tales como poderían ser:

- Navegación en aeroportos ou estacións de metro. As balizas permitirían aos pasaxeiros situarse dentro destes lugares.

- Información en museos. Os visitantes poden obter información de interese baseada na súa localización dentro do museo.
- Xestión de inventarios. Se instalamos unha baliza en cada obxecto poderemos coñecer a súa posición.
- Evacuacións. En grandes instalacións, a xente pode ser equipada cunha tarxeta BLE para que podan ser fácilmente localizados en caso de emerxencia.

1.2 Obxectivos

Este proxecto propón empregar a tecnoloxía Bluetooth Low Energy para determinar as posicións de balizas BLE. Estas balizas emiten mensaxes de xeito periódico, que debemos escanear para comprobar o seu RSSI e poder estimar a distancia á que se encontran. Para facer este escaneamento, necesitaremos empregar certo hardware (rastrexadores) e crear un script que faga o escaneamento.

Mediante os datos escaneados débense calcular as posicións das balizas. Estes datos, e as posicións calculadas, deben ser almacenados para posteriormente poder ser representados. Para procesar os datos dos escaneamentos, necesitaremos crear un servidor de aplicacións que os xestione. As posicións deben escanearse periodicamente, polo que debemos procesar os datos simultaneamente.

Outro obxectivo do proxecto é formar unha rede co número desexado de rastrexadores, que permita escanear as balizas en espazos grandes e complexos e sexa independente doutras redes externas.

Por último, o cálculo das posicións será mediante multilateración, aproveitando deste xeito a rede de rastrexadores dispostos no sistema.

1.3 Proposta

Neste proxecto proponse crear un novo sistema de localización en interiores, co obxectivo de localizar dispositivos BLE, creando unha rede independente e mellorar os resultados da localización mediante o cálculo por multilateración, do que falaremos na sección 3.4.2. Ademais empregarase diferentes tecnoloxías tanto para a recollida e xestión dos datos, como na interface de usuario.

Esta proposta segue a filosofía de utilizar Raspberry como rastrexadores debido ao seu baixo custo, xa que este sistema nace como unha alternativa os sistemas de localización existentes que necesitan unha maior inversión en termos de infrastrutura. Ademais outra das filosofías seguidas é traballar cun plano 2D, co cal poderemos localizar os dispositivos dentro dun mesmo andar.

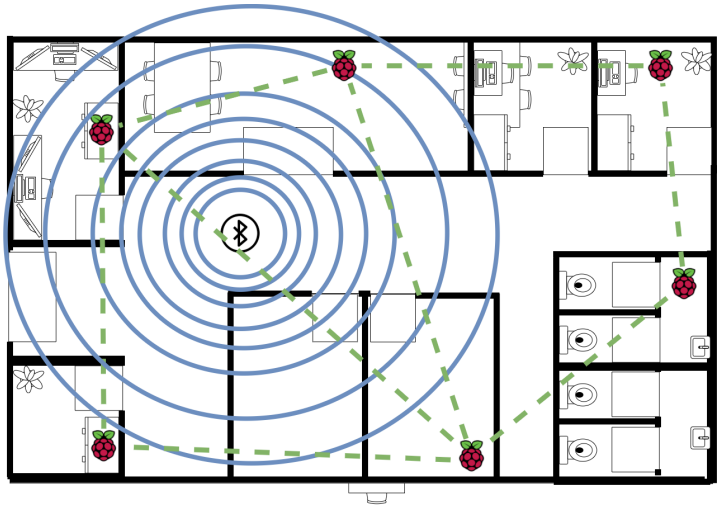


Figura 1.1: Esquema da proposta do sistema de localización a nivel físico.

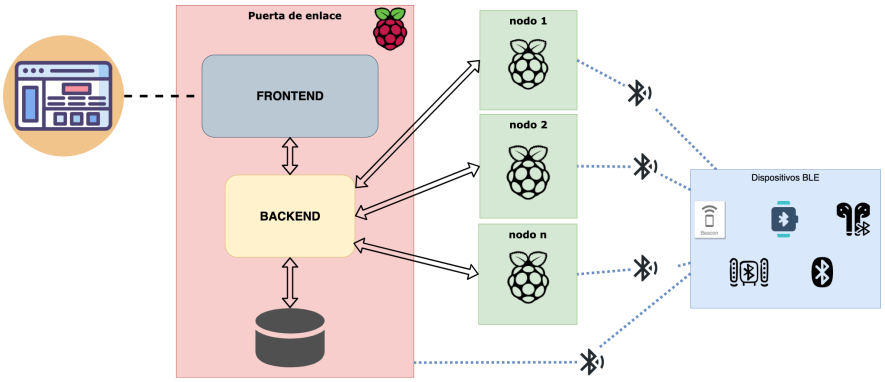


Figura 1.2: Esquema da proposta do sistema de localización a nivel lóxico.

Para o escaneamento das balizas executarase un proceso en segundo plano que recolle a forza dos sinais dos dispositivos de xeito periódico, mentres que para calcular a posición, outro proceso en segundo plano fará os cálculos por multilateración, o cal permitirá ubicar os dispositivos posteriormente.

Os datos do sistema serán accesibles dende unha API REST e dende unha interface donde se poden configurar diferentes parámetros para o escaneamento. Para visualizar as posicións usamos un mapa xerado con Google Maps onde poderemos localizar os dispositivos na infrastrutura na que nos encontremos.

Na imaxe 1.1 observamos como sería a nosa rede no andar dun edificio nun plano 2D. Como vemos, varias Raspberry detectarían unha baliza e comunicaríanse entre sí para recoller o datos e despois calcular a posición.

Na imaxe 1.2 vemos como sería o concepto que propoñemos para a arquitectura de recollida, procesado e visualización dos datos do sistema.

1.4 Estrutura da memoria

Neste apartado resúmese brevemente o contido de cada capítulo da memoria.

- Introdución. Presentación do proxecto, incluíndo obxectivos e propostas.
- Estado do arte. Análise da actualidade sobre temas do proxecto, tecnoloxías e decisión de traballo. de localización, mercado, etc.
- Fundamentos teóricos. Definición e explicación sobre a tecnoloxía Bluetooth, e os algoritmos usados para o cálculo das posicións dos dispositivos BLE.
- Fundamentos tecnolóxicos. Explicación das diferentes tecnoloxías o ferramentas e recursos empregados.
- Análise de requisitos, actores e casos de uso.
- Metodoloxía empregada e a súa adaptación.
- Planificación e custos. Planificación e seguemento do proxecto coa metodoloxía emprega e análise do custo dos recursos empregados.
- Deseño e arquitectura. Decisións estruturais do sistema, da topoloxía e das decisións técnicas empregadas para a creación da aplicación.
- Implementación. Explicación detallada da implementación dalgúns dos elementos máis importantes.
- Probas. Probas realizadas para validar o noso sistema (conexión da rede, independencia do sistema, calibración dos dispositivos, probas de localización das baliza).
- Conclusións. Valoración do proxecto e traballos futuros.
- Apéndice A. Manual de instalación, e de uso do sistema.
- Apéndice B. Código fonte do software empregado.

Capítulo 2

Estado do arte

Neste capítulo expóñense as tecnoloxías máis usadas actualmente para os sistemas de localización en interiores. Tamén falaremos das alternativas de mercado e dos traballos relacionados máis relevantes.

2.1 Posibles tecnoloxías actuais de localización

2.1.1 GPS

Os localizadores por GPS reciben o soporte dunha constelación de 24 satélites, que orbitan por todo o globo terrestre, enviando sinais para quen quera usalos. Un receptor de GPS que quere localizarse no globo terrestre, localiza polo menos a 4 satélites e recibe deles a súa posición e o tempo de envío. Con estes datos o receptor calcula mediante trilateración, a posición do receptor [3]. Como podemos deducir a localización mediante satélites en espazos interiores non é moi boa así que é unha tecnoloxía que podemos descartar para o noso sistema.

2.1.2 RFID

RFID é unha alternativa para a localización en interiores. Este sistema baséase en etiquetas de radiofrecuencia que conteñen unha antena emisora/receptora que ao ser excitada por un transmisor emite un sinal [4]. Tamén existen etiquetas que emiten activamente, pero son máis caras. Así un usuario podería localizarse nun lugar mediante estas etiquetas. O problema que ten este sistema é o seu prezo xa que o alcance desta tecnoloxía non é moi alto (5 metros) [5], polo que se quixésemos implementar este sistema necesitaríamos moitas etiquetas ou moitos receptores, o cal non é interesante.

	Wi-Fi	Bluetooth	Zigbee
Standards	IEEE 802.11.x	IEEE 802.15.1	IEEE 802.15.4
Data rate	11 to 54 Mb s^{-1}	1 Mb s^{-1}	10-115 kb s^{-1}
Latency (time to establish a new link)	< 3 s	< 10 s	30 ms
Frequencies	2.4 and 5 GHz bands	2.4 GHz	2.4 GHz
No. of nodes	> 100	8	65,000
Range	100 m	8 m (Class II, III) to 100 m (Class I)	10-75 m
Modulation	DSSS[1] and OFDM[2]	FHSS[3]	DSSS[1]
Network topology	Star-access point	*Ad hoc* piconets	*Ad hoc*, star, mesh
Data type	Video, audio, graphics, pictures, files	Audio, graphics, pictures, files	Small data packet
Battery life	Hours	1 week	> 1 year
Extendibility	Roaming possible	No	Yes

Figura 2.1: Comparativa estatística entre Wi-Fi, BLE e ZigBee .

2.1.3 Bluetooth

Bluetooth é unha tecnoloxía de comunicación entre dispositivos de curto alcance, de baixa transmisión (Fig. 2.1) e pouco custo. Concretamente a súa versión Bluetooth Low Energy (BLE) é unha das máis empregadas en localización en espazos interiores debido sobre todo a que ten pouco consumo para o alcance que ten.

2.1.4 ZigBee

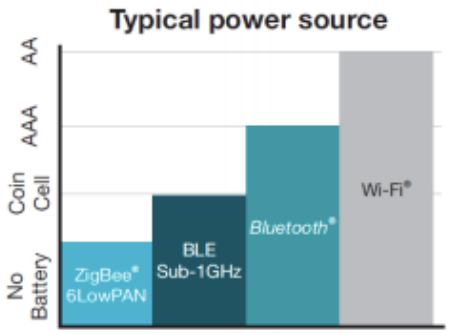
ZigBee é unha tecnoloxía bastante recente, semellante a Bluetooth, de baixo custo e menor consumo [6]. Este protocolo está máis pensado para facer mallas entre os dispostivos, pero por contra é unha tecnoloxía de menor velocidade (Fig. 2.1) e todavía está en pañais. No futuro pode ser unhas das tecnoloxías máis destacadas en localizacións en interiores.

2.1.5 Wi-Fi

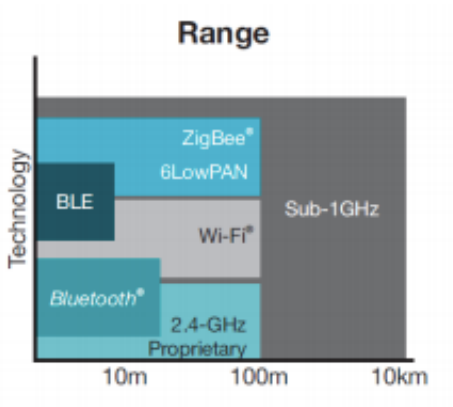
A tecnoloxía Wi-Fi 802.11 é o estándar de comunicación sen fíos máis recoñecido globalmente. Este protocolo ten varias vantaxes fronte a outras tecnoloxías como que ten un bo alcance, unha boa velocidade de transmisión (Fig. 2.1) , está implantado mundialmente e non necesita demasiados puntos de acceso [7]. Por contra ten algunas desvantaxes como que consume máis potencia ou que é máis inestable. Existe unha versión específica para localización en interiores, pero case non hai hardware comercializado.

2.2 Comparativa e decisión de traballo

Despois de ver algunhas das tecnoloxías máis empregadas para a localización debemos tomar a decisión de cal temos que escoller para o noso sistema. A localización GPS foi descar-



(a) Consumo das tecnoloxías ZigBee, Bluetooth e Wi-fi.



(b) Rango das tecnoloxías ZigBee, Bluetooth e Wi-Fi.

Figura 2.2: Comparativa gráfica entre Wi-Fi, BLE e ZigBee.

tada sen moito reparo debido aos motivos obvios da súa problemática. A localización mediante RFID podería resultar interesante a nivel de precisión pero o seu rango (aprox. 3 metros de media) obrigaría a un despregamento de infraestrutura moi grande. Polo tanto quedan 3 posibilidades sobre as que debater. Para facer unha comparativa na imaxe 2.1 [8] amósanse as diferenzas entre elas.

Como podemos observar aínda que Wi-Fi e ZigBee teñen o maior rango temos que descartalas polas seguintes razóns:

- Aínda que ten a vantaxe de que necesitaría menos puntos de acceso debido o seu maior rango, o consumo de Wi-Fi é moi elevado respecto as demais. Como podemos observar as baterías durarían poucas horas e isto faría aumentar os gastos. (Máis consumo = máis gasto)

- ZigBee aínda que é a que mellores parámetros ten para o sistema en cuestión, é un sistema con baixa implantación no mercado. Nun futuro pode ser unha das tecnoloxías máis punteiras neste sector, pero mentres non haxa un maior traballo con este estándar é un sistema moi díficil de implementar: Máis tempo de desenvolvemento = máis gasto.

Ademais destes problemas as vantaxes que ten Bluetooth, como que agora mesmo existen infinidade de dispositivos que xa o teñen integrado polo que poderían valer para o noso sistema, ou que a súa versión Bluetooth Low Energy ten un consumo bastante baixo.

Para unha comparativa máis visual amósanse as figuras 2.2a e 2.2b [9] onde podemos ver gráficas sobre os consumos e rangos destas tecnoloxías.

2.3 Traballos e estudos relacionados

Como base para o noso proxecto estudáronse algúns traballos e estudos previos. O artigo *A Bluetooth Low Energy Indoor Positioning System with Channel Diversity, Weighted Trilateration and Kalman Filtering* [10], fai referencia a un estudio semellante no que analizan un sistema de localización mediante Raspberry. Neste artigo analizan entornos de diferentes tamaños, nos que despregan sistemas de localización con rastrexadores. Este artigo será interesante para comparar os resultados do noso sistema co estado da arte.

Outro traballo consultado para analizar o sistema é o TFG de *Val Jímenez* [11], no que propón un sistema de localización en interiores empregando RSSI, pero a diferencia do noso sistema, emprega outros algoritmos de cálculo da posición como son o método de interpolación ou o máximo RSSI. Este traballo serviranos para analizar o proceso de escaneamento das balizas BLE.

2.4 Mercado

Algúns dos sistemas de localización en espazos interiores máis recoñecidos no mercado actual son os seguintes:

- Atria [12]. Esta empresa afirma ter múltiples aplicacións de localización en interiores tales como:
  - Localización de activos, ferramentas, equipos e productos. Para coñocer en todo momento donde se están ou amosar a súa trazabilidade.
  - Seguridade. Evitar que persoas se acheguen a zonas perigosas.
- Situm [13] Sistema baseado tanto en tecnoloxía Wi-Fi coma en Bluetooth. Funciona para localizar smartphones cunha mínima infraestrutura e con bastante precisión.
- BlueRange [14]. Mediante balizas pegadas aos obxectos este sistema permite a localización empregando as balizas pegadas coma unha rede interconectada. Proporciona tamén localización en tempo real.

Capítulo 3

Fundamentos teóricos

PARA entender o noso proxecto é importante coñecer o que é a tecnoloxía Bluetooth Low Energy, máis coñecida como BLE. A continuación coméntase os aspectos teóricos sobre ela.

3.1 Concepto BLE

Bluetooth Low Energy é unha tecnoloxía para a transmisión de datos sen fíos en redes de curto alcance [15]. Nace coma unha variante máis lixeira do Bluetooth clásico e ten coma característica máis importante o seu baixo consumo de enerxía e ao igual que o Bluetooth clásico, emprega a banda de 2.4 GHz para transmitir. Táboa 3.1

Táboa 3.1: Diferenzas entre Bluetooth clásico e BLE.

Características	Bluetooth Clásico	BLE
Consumo	<30mA	<15mA
Trasnferencia de datos	2-3 Mbps	200 kbps
Alcance	<30 metros	50 metros en espazos abertos

O uso desta tecnoloxía está máis orientado a dispositivos usados en IoT debido a que son dispositivos que non teñen a necesidade de usar tanta potencia o que implicaría un maior consumo.

3.2 Conceptos da comunicación entre dispositivos BLE

Existen dous tipos de dispositivos BLE [16], os periféricos, que son pequenos dispositivos con poucos recursos e polo tanto baixa potencia; e os centrais, que son dispositivos que teñen máis recursos e capacidade de procesamento.

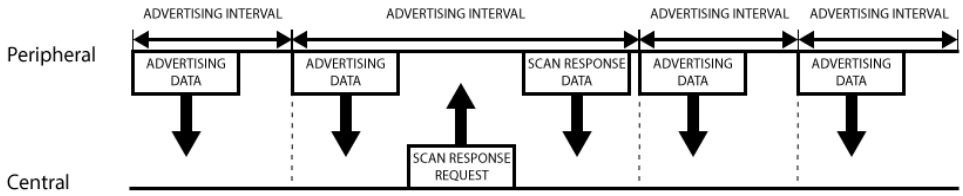
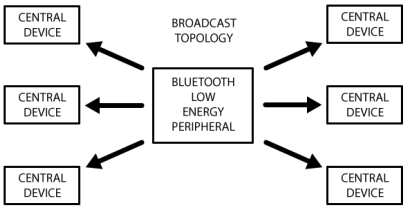
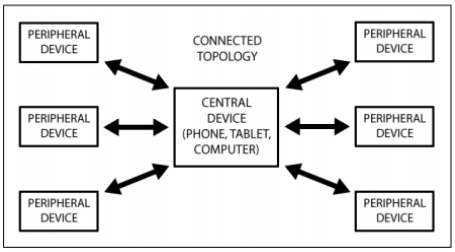


Figura 3.1: Proceso de **advertisement** e **broadcast** en dispositivos BLE.



(a) **Broadcasting** periférico-centrais.



(b) Conexión **punto a punto** central-periférico.

Figura 3.2: Comunicación entre dispositivos BLE.

Para a comunicación, os periféricos estarán emitindo paquetes de xeito regular para poderen ser descubertos, o que se coñece como **advertisement** (ver Fig. 3.1 [1]), ademais tamén responderán as posibles peticións de escaneamento para obter máis información delas, o que se adoita chamar **scan response**. Os dispositivos centrais escanean os paquetes que se emiten regularmente e se necesitan máis información dos periféricos mandan unha petición.

Estes dous tipo de mensaxes implican dous tipos de comunicación:

- Comunicación **broadcast**: é o proceso polo que os dispositivos periféricos mandan os paquetes regularmente para ser descubertos polos dispositivos centrais. Na figura 3.2a [1] podemos velo.
- Comunicación **punto a punto**: é o proceso polo cal un dispositivo central conéctase a un periférico para a transmisión de datos. Na figura 3.2b [1].

3.3 Indicador de forza do sinal recibido (RSSI)

O indicador de forza do sinal recibido (RSSI) é unha medida estimada do ben que un dispositivo pode oír, detectar ou recibir sinais de calquera punto de acceso. A idea do RSSI é que axuda a determinar se un sinal ten a suficiente forza coma para establecer unha conexión inalámbrica [17]. A medida ca distancia aumenta, o sinal debilitase e o ancho de banda da conexión de datos sen fíos faise lento. Este RSSI adoita ser invisible para o usuario dun dispositivo receptor, pero coma a intensidade do sinal varía enormemente e afecta á función dunha

conexión inalámbrica, os dispositivos a veces poñen a medida a disposición dos usuarios. En poucas palabras, o RSSI é nome máis común para referirse á forza coa que un dispositivo está escoitando a outro.

O RSSI exprésase en dBm que é unha unidade de medida de potencia expresada en decibelios (dB) relativa a 1 milivatio (mW). Este valor de RSSI varía nunha escala logarítmica entre 0 e -100, donde 0 é unha intensidade alta e -100 practicamente nula. Este valor pode verse afectado por diversos factores ambientais como pode ser a temperatura a humidade, etc. así como o ruído doutros sinais ou obstáculos (paredes, persoas, tipo e orientación da antena, electrónica de amplificación empregada, etc.). O valor tamén pode variar entre dous dispositivos á mesma distancia, dependendo da potencia ou a frecuencia configurada.

3.4 Cálculo da posición a partir do RSSI

Como acabamos de ver, existe unha relación entre a distancia e o RSSI, pola cal mediante lecturas periódicas do RSSI poderemos estimar a posición dun dispostivo. Existen diferentes xeitos de facelo, pero en todos existe unha relación entre o RSSI e a distancia. No noso caso empregamos a fórmula [18] [19]:

$$Distancia = 10^{((PotenciaMedia - RSSI)/(10*N))} \tag{3.1}$$

Donde:

- Distancia é a distancia medida en metros.
- A potencia media é unha constante calibrada de fábrica en cada dispositivo que indica cal é o RSSI esperado a unha distancia de 1 metro da baliza. Combinado co RSSI, permite estimar a distancia. Este valor pode ser modificado se tes acceso aos comandos internos do chip BLE. Como comentamos antes, hai outro factor que tamén afecta ao sinal continuo que é o intervalo de publicación (*advertisement* (ver Sec. 3.2)):
  - As balizas non emiten constantemente como Wi-Fi. No seu lugar, parpadean. O intervalo de publicación describe o tempo entre cada parpadeo. O valor oscila entre 100 ms e 2000 ms. É preciso ter en conta que reducir o intervalo de publicación repercutirá na vida da batería.
- N é a constante que depende do factor ambiental. Oscila nun rango entre 2 e 4, sendo 2 para sistemas poucos conxestionados e 4 para sistemas moi conxestionados.
- RSSI é o parámetro chave neste cálculo xa que é o indicador de intensidade do sinal recibido (ver Sec. 3.3). É a forza do sinal da baliza que se ve no dispositivo receptor.

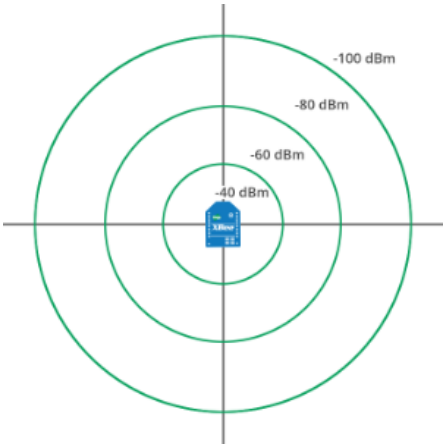


Figura 3.3: Variación do RSSI coa distancia de xeito radial.

> Debido a factores externos que inflúen nas ondas de radio, como a absorción, a interferencia ou a difracción, o RSSI tende a flutuar. Canto máis lonxe estea o dispositivo da baliza, máis inestable se fai o RSSI.

O artigo [10], fala de 3 modelos diferentes para estimación da distancia fronte o RSSI. Como podemos comprobar, a forza do sinal sofre unha perda logarítmica a medida que aumenta a distancia, como no noso caso.

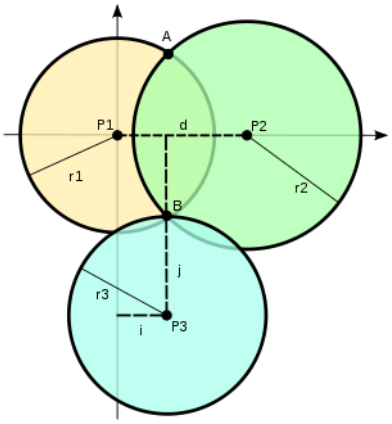
A fórmula empregada permítenos calcular a distancia entre a baliza e un dispositivo que recibe todas esas mensaxes periódicas. A este dispositivo chamarémoslle rastrexador. Segundo cantos rastrexadores teñan información sobre o RSSI destas mensaxes, poderemos calcular a posición de varios xeitos: mediante o radio da distancia (1 rastrexador detecta a baliza) e mediante multilateración (máis de 1 rastrexador detecta a baliza).

3.4.1 Un rastrexador | Radio da distancia

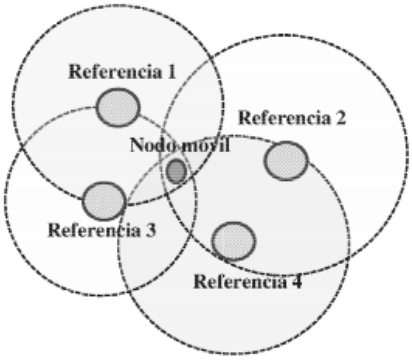
Este é un método de baixa precisión, xa que se basea na posición do rastrexador. Este rastrexador escanea a baliza e mediante o cálculo da distancia co RSSI pode estimar un radio no que a baliza podería estar ubicada. As limitacións deste cálculo débense a que sabemos a distancia á que está, pero non sabemos en que dirección, polo que estimar un radio parece a mellor opción. Este método é semellante o que empregan algúns GPS cando non poden concretar unha posición. Na imaxe 3.3 [20] podemos observar o concepto.

3.4.2 Máis dun rastrexador | Multilateración

O funcionamento normal do noso sistema baséase no concepto de multilateración [10]. Para entender o que é a multilateración é importante coñecer primeiro a trilateración xa que



(a) Esquema teórico da trilalateración [22].



(b) Esquema real da multilateración [23]

Figura 3.4: Concepto de trilateración e multilateración.

é unha evolución desta. A trilateración é un método matemático para determinar as posicións relativas dos obxectos usando a xeometría do triángulo. A trilateración usa as localizacións coñecidas de dous ou máis puntos de referencia e a distancia medida entre o suxeito e cada punto. Para determinar de xeito único e preciso a situación relativa dun punto nun plano bidimensional usando só a trilateración, polo xeral son necesarios polo menos 3 puntos de referencia como podemos ver na imaxe 3.4a.

Algo que debemos ter en conta é que isto é un modelo matemático e na realidade as lecturas do RSSI, como xa explicamos anteriormente (ver Sec.3.3), non son constantes debido aos factores ambientais, polo que os radios de distancia non se cortaran á perfección, como no modelo teórico, e haberá que estimar a posición no rango posible. Ante esta situación xorde a multilateración que aporta máis de 3 puntos de referencia polo que deste modo poderemos reducir o erro ao producirse máis puntos de corte como vemos na imaxe 3.4b Este modelo pódese calcular coa librería de Python *Localization* [21]

Capítulo 4

Fundamentos tecnolóxicos

Neste capítulo exponse as tecnoloxías, ferramentas e recursos empregados neste proxecto ademais das ferramentas para a edición e xestión.

Estas son as tecnoloxías e ferramentas usadas e o motivo da súa elección.

4.1 Tecnoloxía na lóxica de negocio

A lóxica de negocio son rutinas que realizan entradas ou consultas aos datos, xeración de informes e máis especificamente todo o procesamento que se realiza detrás da aplicación visible para o usuario. Adóitase chamar BackEnd.

No noso caso optouse por realizar unha API REST conectada a unha base de datos onde se gardarán os datos. Unha API é un conxunto de definicións e protocolos que se usan para deseñar e integrar o software de aplicaciones. Adoita considerarse como o contrato entre un proveedor de información e un usuario, donde se establece o contido que se require do consumidor (chamada) e o que necesita o produtor (a resposta) [24].

O concepto de REST é un conxunto de principios para arquitectura da aplicación. Para que unha aplicación sexa REST debe cumprir o seguinte:

- A Arquitectura cliente-servidor está composta de clientes, servidores e recursos, coa xestión de solicitudes a través de HTTP.
- A Comunicación entre o cliente e o servidor é sen estado, así que non se almacena a información do cliente entre as solicitudes; cada unha é independente e está desconectada do resto.
- Datos que poden almacenarse en caché e optimizan as interaccións entre cliente e servidor.
- A información entrégase por medio de HTTP nalgún destes formatos: JSON, HTML ou texto sen formato.

4.1.1 Framework empregado

Para traballar con este servidor de aplicacións optouse por Django [25]. Django é un framework de código aberto web Python de alto nivel que fomenta o desenvolvemento rápido cun deseño limpo e pragmático. Ademais é facilmente escalable e permite a integración con outras tecnoloxías para ter unha aplicación máis modular, como por exemplo múltiples bases de datos. Neste proxecto, Django encargarase de levantar a API REST a partir do modelo de datos, para que a interface de usuario poda consultar a información relevante.

4.2 Base de datos

A base de datos empregada é SQLite3. SQLite é un sistema de xestión de bases de datos relacional compatible con ACID, contida nunha pequena biblioteca escrita en C. SQLite é un proxecto de dominio público e é a base de datos máis utilizada no mundo [26]. O motivo de escoller esta base de datos é porque é pequena, rápida, autónoma, de alta confiabilidade e moi portátil. Ademais a súa sinxeleza para a velocidade de desenvolmento da aplicación, tamén é un punto moi forte na decisión.

A base de datos está integrada no framework de Django que comentamos con anterioridade na sección 4.1.1.

4.3 Rastrexadores e balizas BLE

Para a lectura das balizas BLE necesitaremos dispositivos que fagan esta labor e ademais que teñan capacidade de procesar a información recollida. Para este proxecto optouse por unha opción de baixo custo como son as Raspberry. Na imaxe 4.1 [27] podemos ver as especificacións destes dispositivos. No noso caso empregáronse tres Raspberry Pi 3 e catro Raspberry Pi Zero W. A Raspberry Pi 3 é o modelo estándar, e a Raspberry Pi Zero W, é unha versión de todavía menor custo, con capacidade de conexión Wi-Fi e que consume menos enerxía.

A elección deste hardware ven motivada polo seu prezo en comparación con outros rastrexadores comerciais e a liberdade que proporciona un sistema operativo baseado en Linux. Neste caso o S.O. é Raspbian [28].

4.3.1 Librerías empregadas

Para poder detectar as diferenzas de RSSI na nosa aplicación usaremos a librería para BLE, *Bluepy* [29]. Esta librería é un módulo de Python, basado no proxecto *Bluez*, que permite a comunicación con dispositivos Bluetooth Low Energy (BLE). Utilízase para monitorizar os

Comparativa Raspberry Pi

comohacer.eu ¿inventamos juntos?

	SoC	CPU	GPU	RAM	USB	V/A	Boot	Red	Alimentación	Tamaño	Fecha	Precio
3 Model B	Broadcom BCM2837	1,2GHz QUAD ARM Cortex-A53	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100 WiFi, BT	2,5A 12,5w / 5v MicroUSB GPIO	85 x 56 mm	02/16	35$
3 Model B+	Broadcom BCM2837B0	1,4GHz QUAD ARM Cortex-A53	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100/300 (USB) Dual-band WiFi BT	2,5A 12,5w / 5v MicroUSB GPIO PoE (HAT)	85 x 56 mm	03/18	35$
4 Model B	Broadcom BCM2711	1,5GHz QUAD ARM Cortex-A72	VideoCore IV	1, 2 o 4GB	2 (2.0) 2 (3.0)	Jack 2 micro HDMI	uSD	ETH 1000 Dual-band WiFi BT	2,5A 12,5w / 5v USB-C GPIO PoE (HAT)	85 x 56 mm	06/19	35$
Zero	Broadcom BCM2835	1GHz ARM1176JZF-S	VideoCore IV	512MB	1 Micro	Mini HDMI	uSD	No	160mA 0,8w / 5v MicroUSB GPIO	65 x 30 mm	11/15	5$
Zero W	Broadcom BCM2835	1GHz ARM1176JZF-S	VideoCore IV	512MB	1 Micro	Mini HDMI	uSD	Wifi, BT	160mA 0,8w / 5v MicroUSB GPIO	65 x 30 mm	02/17	10$

Figura 4.1: Especificacións das Raspberry Pi 3 e Zero W.

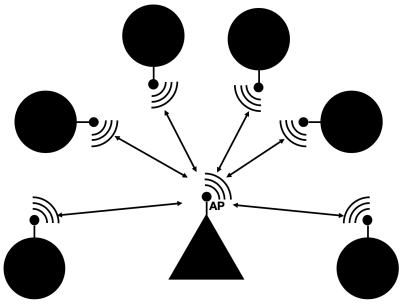
Figura 4.2: Baliza BLE modelo IBKS 105.

dispositivos BLE que no noso caso para as probas serán os IBKS 105 que podemos ver na imaxe 4.2 [30] e que podemos consultar as súas especifiacións no seu *datasheet* [31].

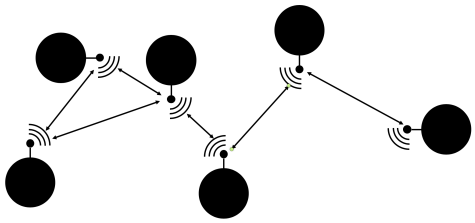
Para o cálculo da posición, como tratamos na sección 3.4.2, usaremos a librería *Localization* xa mencionada anteriormente [21].

4.4 Comunicación entre rastrexadores

O noso sistema conta con múltiples rastrexadores polo que debemos buscar un xeito de conectalos para poder recoller os datos que proporcionan. Para iso debemos empregar algún tipo de rede. O patrón máis común para as redes sen fíos é a topoloxía en estrela, onde hai un punto de acceso central ou estación emisora e todos os dispositivos que queiran comunicarse na rede deben estar ao alcance do punto de acceso. Toda a comunicación na rede ocorre entre un dispositivo e o punto de acceso, polo que o tráfico entre 2 dispositivos da rede debe pasar polo punto de acceso. Se un dispositivo móvese pode desconectarse dun punto de acceso e conectarse a un punto de acceso diferente para manter a conectividade, pero para estar na



(a) Topoloxía en estrela.



(b) Topoloxía malla.

Figura 4.3: Topoloxías de comunicación nunha rede.

rede ten que estar ao alcance dun punto de acceso.

Cunha rede de malla non hai ningún punto de acceso central. A rede está formada polos dispositivos que realizan conexións directas con outros dispositivos do alcance (veciños). O protocolo de malla e os algoritmos que xestionan a rede traballan como enrutar paquetes de datos, polo que calquera dispositivo que forma parte da rede pode falar con todos os demais dispositivos da rede, sempre que haxa unha ruta dispoñible a través da rede.

Se quixésemos empregar Internet, coa topoloxía en estrela, o dispositivo debería ter conexión directa co punto de acceso que debe estar conectado a unha rede con acceso a internet. Cunha rede de malla, calquera dos dispositivos de malla pode ser unha porta de entrada a outra rede con conectividade a Internet.

Polas vantaxes que ten unha rede malla para este tipo de aplicación, que nos permitirían prescindir de dispositivos externos sobre o que montar a noso sistema, e posibilidade de aumentar exponencialmente a nosa rede, decidimos que é a mellor opción.

Tecnoloxía empregada

Para xestionar a rede de malla, cómpre instalar unha utilidade chamada *batctl* que é a ferramenta de configuración e depuración de *batman-adv* [32]. *Batman-adv* é unha utilidade que permite por en marcha unha red de malla [33].

4.4.1 Interface de usuario

O obxectivo da interface gráfica é expoñer os datos proporcionados pola Application Programming Interface Representational State Transfer (API REST) aos usuarios dun xeito amigable, que permita consultar os datos, usar as utilidades ou configurar o sistema de xeito sinxelo.

Framework empregado

Para esta interface empregouse Vue.js. Vue.js é un marco progresivo para construír interfaces de usuario. A diferenza doutros cadros monolíticos, Vue.js está deseñado dende cero para ser usado de forma incremental. A biblioteca central céntrase só na capa de visualización e é fácil de usar e integrar con outras bibliotecas ou proxectos existentes. Por outra banda, Vue tamén é perfectamente capaz de dirixir aplicacións sofisticadas de páxina única cando se usa en combinación con ferramentas modernas e bibliotecas compatibles. [34]

4.4.2 Servizos

A nosa aplicación debe estar monitorizando os dispositivos BLE, mediante a librería *Bluepy* mencionada anteriormente na sección 1.1. Para estar monitorizando isto, debemos ter algún tipo de proceso en segundo plano que empregue esta librería é comunique os cambios de RSSI a base de datos. Ademais debemos ter un segundo proceso que consuma os datos de todos estes escaneamentos e calcule a posición dos dispositivos en cada instante de tempo.

Tecnoloxía empregada

Para estes dous procesos utilizarase dous scripts de Python que se executarán como servizo do sistema unha vez se acenda o rastrexador. Para comunicarse coa base de datos empregarase o protocolo MQTT.

MQTT é un protocolo de comunicación M2M (máquina a máquina) do tipo de cola de mensaxes [35]. Baséase na pila TCP/IP como base para a comunicación. No caso de MQTT, cada conexión mantense aberta e reutilízase en cada comunicación, así diferénciase, por exemplo, dunha solicitude HTTP 1.0 onde cada transmisión realízase a través da conexión. Funciona como un servizo de mensaxería *push* cun patrón de publicador/subscritor (pub-sub) onde os clientes conéctanse a un servidor central chamado *broker* e para filtrar as mensaxes que se envían a cada cliente, as mensaxes dispóñense en temas organizados xerarquicamente. Un cliente pode publicar unha mensaxe sobre un tema determinado. Outros clientes poden subscribirse a este tema e o corredor enviaralle as mensaxes subscritas. Na imaxe 4.4 [35] podemos velo graficamente.

4.4.3 Ferramentas e utilidades de desenvolvemento

Nesta sección inclúese todas as ferramentas que permitiron desenrolar o proxecto e documentalo.

- Git e GitLab. Sistema de control de versións distribuído de código aberto. As súas principais características son a velocidade, e un deseño sinxelo. GitLab será o servidor

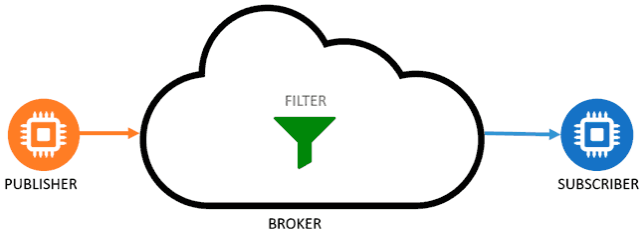


Figura 4.4: Protocolo MQTT

remoto para o sistema de control de versións.

- Pycharm. Contorna de desenvolvemento integrado (IDE) que se emprega en programación, especificamente para a linguaxe Python e proporciona análise de código, un depurador gráfico, un probador de unidades integrado, integración con sistemas de control de versións e admite o desenvolvemento con Django.
- Visual Studio Code. Editor de código fonte que inclúe soporte para a depuración, control integrado de Git, resaltado de sintaxe, finalización intelixente de código, fragmentos e refactorización de código. É de código aberto e ademais baséase en Electron, un framework que se utiliza para implementar Chromium e Node.js como aplicacións para escritorio.
- LaTex e Overleaf. LaTex é un sistema de composición de textos, orientado á creación de documentos escritos que presenten unha alta calidade tipográfica. Polas súas características e posibilidades, é usado de forma especialmente intensa na xeración de artigos e libros científicos que inclúen, entre outros elementos, expresións matemáticas. Overleaf é un editor colaborativo de LaTeX baseado na nube que se utiliza para escribir, editar e publicar documentos científicos de forma colaborativa.
- Postman Desktop. Ferramenta que permite realizar peticións HTTP. Esta aplicación é útil no desenvolvemento de servizos REST permitindo realizar tests.
- GoogleColab. Entorno de máquinas virtuales basado en Jupyter Notebooks no que realizamos gráficas para a documentación.
- Draw.io é un software de diagramas en liña gratuíto para crear diagramas de fluxo, diagramas de procesos, organigramas, UML, entidade-relación e diagramas de rede.
- Librerías para cálculos matemáticos: numpy, scipy, etc. Bibliotecas para a linguaxe de programación Python que dan soporte para crear vectores e matrices grandes multidimensionales, xunto cunha gran colección de funcións matemáticas de alto nivel para operar con elas.

Capítulo 5

Análise de requisitos

Neste capítulo analízanse os requisitos do proxecto. Os requisitos xorden no proceso de estudo das necesidades dos usuarios tanto a nivel hardware e software.

5.1 Requisitos funcionais

Podemos resumir os principais requisitos funcionais do proxecto nos seguintes:

- Escaneamento dos dispositivos BLE. Trátase da obtención das mensaxes periódicas que mandan as balizas para comprobar o seu RSSI a través do hardware necesario (rastrexadores).
- Almacenamento dos datos escaneados. É importante, unha vez escaneados as mensaxes dos dispositivos BLE, gárdalas para procesalas posteriormente.
- Cálculo das posicións. Nun sistema de localización sobra decir que o cálculo da posición é un requisito indispensable.
- Representación das posicións. Consiste na representación gráfica das posicións das balizas tanto en tempo real como cun histórico.
- Representación dos rastrexadores. Os rastrexadores deben ser tamén representados para poder comparar a súa posición cas balizas.
- Xestión das balizas e rastrexadores. Permitir crear, ver, eliminar e editar os parámetros dos rastrexadores, incluíndo a súa posición.

5.2 Requisitos non funcionais

Podemos resumir os principais requisitos non funcionais do proxecto nos seguintes:

- Usabilidade. O sistema debe ser usado polos usuarios con efectividade, eficiencia, satisfacción e sen necesidade dunha formación específica.
- Escalabilidade. O sistema debe ter a capacidade de adaptación e resposta en termos de rendemento a medida que aumentan de forma significativa o número de usuarios ou recursos empregados.
- Independencia. O sistema debe depender o menos posible de redes externas ou recursos externos.
- Modularidade. O sistema debe estar subdividido en diferentes partes que traballen en funcionalidades independentes e que sexan facilmente sustituíbles sen ter que cambiar todo o sistema.
- Económico. O sistema debe ser de baixo custo ou ser máis barato que as alternativas de mercado.
- Tempo real. O sistema debe ofrecer as posición co mínimo intervalo posible de tempo.

5.3 Actores

Representan os elementos que conflúen no sistema:

- Usuario. Representa a toda persoa que emprega a interface gráfica para consultar as utilidades do sistema.
- Rastrexador. Encárgase de xestionar os datos e procesalos.
- Servizo escaneamento. Forma parte do rastrexador como un proceso en segundo plano que lee os mensaxes das balizas.
- Servizo cálculo posición. Forma parte do rastrexador como un servizo en segundo plano que vai calculando a localización das balizas cos datos dos escaneamentos.

5.4 Casos de uso

5.4.1 Usuario

CU-U1 Xestión das balizas

Este caso consiste na xestión das balizas. Este caso inclúe os subcasos engadir, editar e eliminar. O usuario encárgase da xestión dos datos na interface para comunicarllo o rastrexador.

CU-U2 Xestión dos rastrexadores

Este caso consiste na xestión dos parámetros dos rastrexadores. Este caso inclúe os subcasos engadir, editar e eliminar. O usuario encárgase da xestión dos datos na interface para comunicarllo ao rastrexador.

CU-U3 Xestión da configuración

Neste caso o usuario encárgase de escoller os paramétros internos do sistema.

CU-U4 Visualización das posicións

O usuario solicita a representación gráfica das posicións das balizas. Para eso é necesario ter a representación do histórico ou en tempo real.

5.4.2 Rastrexador

CU-R1 Xestión das balizas

Este caso consiste na xestión das balizas. Este caso inclúe os subcasos engadir, editar e eliminar. O rastrexador recolle os datos ou as peticións proporcionados polo usuario mediante a interface, para xestionar o modelos da baliza na base de datos.

CU-R2 Xestión dos rastrexadores

Este caso consiste na xestión dos parámetros dos rastrexadores. Este caso inclúe os subcasos engadir, editar e eliminar. O rastrexador recolle os datos ou as peticións proporcionados polo usuario mediante a interface, para xestionar o modelos do rastrexador na base de datos.

CU-R3 Xestión da configuración

Neste caso o rastrexador recibe os parámetros de configuración e modifica os arquivos de configuración.

CU-R3 Xestión dos escaneamentos

Este caso consiste na xestión dos escaneamentos. Este caso inclúe o subcaso engadir. O rastrexador recolle os datos comunicados polo servizo de escaneamento e xestiona o modelo do escaneamento na base de datos.

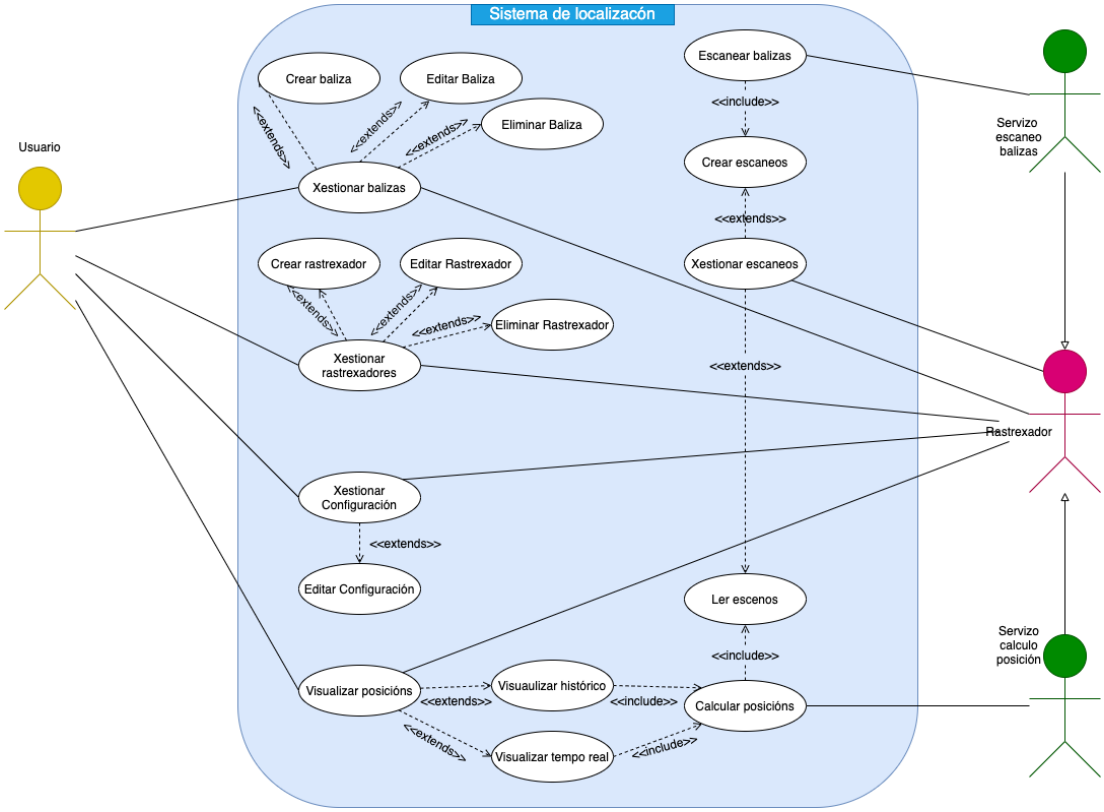


Figura 5.1: Diagrama de casos de uso para o sistema de localización proposto.

CU-R4 Visualización das posicións

O rastrexador recibe a solicitude do usuario, para a representación gráfica das posicións das balizas. Para iso é necesario ter as representación do histórico e en tempo real.

5.4.3 Servizo escaneamento

CU-SE1 Escanear balizas

O servizo de escaneamento lee os cambio de RSSI das balizas e comunícallo o rastrexador.

5.4.4 Servizo cálculo da posición

CU-SP1 Cálculo das posicións

O servizo de cálculo da posición calcula as posicións usando os escaneamentos e comunícallo o rastrexador.

Capítulo 6

Metodoloxía de desenvolvemento

Neste capítulo explícase a metodoloxía escollida para ao proxecto, e como se adaptou ao contexto do proxecto. facendo referencia o conxunto de procedementos racionais empregados para alcanzar o obxectivo.

6.1 Metodoloxía de desenvolvemento áxil | SCRUM

Por definición, as metodoloxías áxiles son aquelas que permiten adaptar a forma de traballo ás condicións do proxecto, conseguindo flexibilidade e inmediatez na resposta para amoldar o proxecto e o seu desenvolvemento ás circunstancias específicas do entorno.

As metodoloxías áxiles baséanse nun modelo incremental e iterativo, no cal o proxecto concíbese como un conxunto de fitos o entregables, que se van desenvolvendo e entregando de forma periódica, o que permite por un lado, adaptar ou reconducir o proxecto en función do desempeño en cada momento e por outro, dispor dun producto funcional en etapas moi tempranas do desenvolvemento.

SCRUM é unha das metodoloxías áxiles máis usada, como se amosa na imaxe 6.1 [36]. Como metodoloxía áxil, consiste en dividir o proxecto nunha serie de fitos ou entregables, de modo que coa entrega e posta en producción de cada un deles, pódese revisar ou redefinir o conxunto do proxecto. Para isto, ao inicio do proxecto defínese unha lista de requisitos ou obxectivos que se deben cumprir, e priorízanse en función do valor que poden aportar, o do retorno da inversión en función do custo e tempo de desenvolvemento. Estas tarefas priorizadas forman o denominado *backlog*.

A partir das tarefas priorizadas formamos os chamados *sprints*, que marcan as tarefas que debemos realizar para cumprir o obxectivo dese *sprint* que dura normalmente 2 semanas. Unha vez acabado o *sprint* realizamos unha *sprint review* na que demostramos os obxectivos acadados e se ocorre algún problema ou xorde unha nova necesidade, podemos crear unha nova tarefa no *backlog* o que nos permite resolver incidencias dunha manera áxil é poder obter

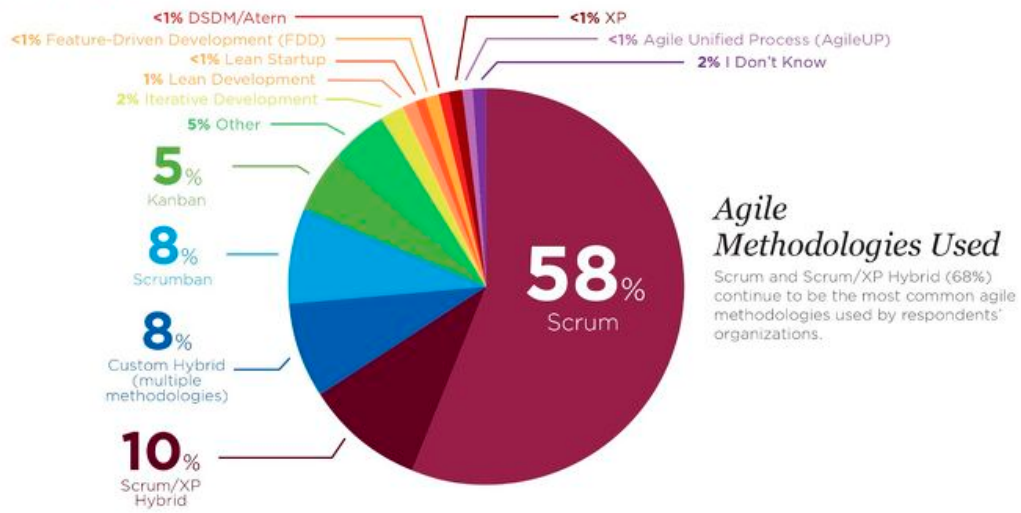


Figura 6.1: Metodoloxías áxiles máis empregadas no proceso de desevenvolmento dun proxecto.

un producto minimamente viable no plazo establecido de tempo. Hai que ter en conta que a metodoloxía SCRUM non fai sinxela a predición de tempo porque se asemella máis a realidade na que realmente no sabes canto tempo pode levantar facer algunha tarefa, pero mediante a estimación dos puntos de esforzo e a experiencia podes planificar aproximadamente o tempo ou o numero de *sprints*. Na imaxe 6.2 [37] podemos ver como é este proceso.

6.1.1 Vantaxes

- Flexibilidade fronte aos cambios. Adáptase ás necesidade mediante a posta en marcha de forma modular ou incremental das funcionalidades.

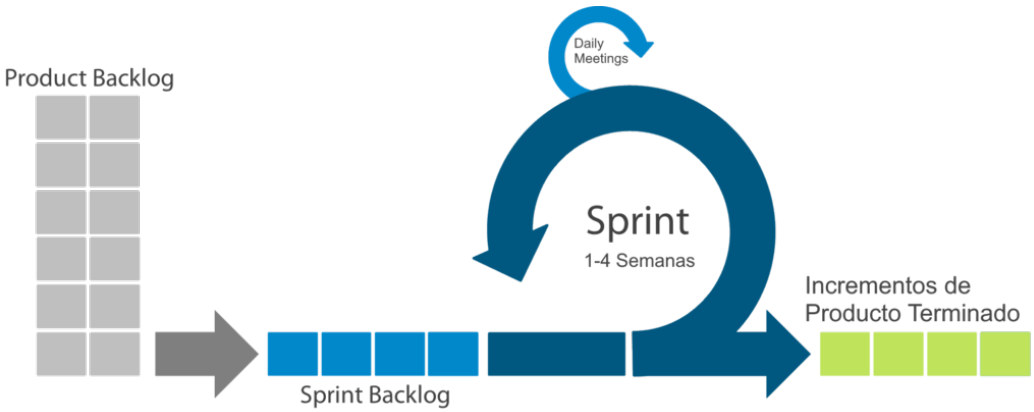


Figura 6.2: Diagrama do proceso da metodoloxía de traballo de desenvolvemento SCRUM.

- Menor tempo mínimo. O feito de que as iteracións formúlense de modo que sexan funcionais e se priorice en función do valor que poidan aportar, resulta en que é posible dispor dun producto minimamente viable (MVP) nunha etapa temprana.
- Maior calidade do software. A aproximación ao desenvolvemento de forma incremental e a obriga de planificar as entregas dun xeito funcional axudan a mellorar a calidade do software, o que por outra banda, ao pasar revisións constantes e demostracións, permite detectar posibles erros ou ineficiencias.
- Maior productividade. A organización en grupos autosuficientes e autónomos do equipo de traballo permite reducir a burocracia e impedimentos no desenvolvemento das iteracións, o que axiliza a toma de decisións e a resolución de problemas.
- Menor risco. Validar as funcionalidades chave do proxecto nas fases iniciais permite validar a viabilidade do proxecto sen necesidade de realizar o investimento que suporía o desenvolvemento completo, así como avaliar a calidade do software.

6.1.2 Desvantaxes

- Scrum a miúdo leva ao aumento do alcance, debido á falta dunha data de finalización definitiva.
- As posibilidades de fracaso do proxecto son altas se as persoas non están moi comprometidas e cooperan.
- Adoptar o marco Scrum en equipos grandes é un reto.
- As reunións diarias ás veces frustran aos membros do equipo.
- Se algún membro do equipo sae no medio dun proxecto, pode ter un enorme impacto negativo no proxecto.
- A calidade é difícil de implementar ata que o equipo pasa por un proceso de proba agresivo.

6.2 Implantación no proxecto

No noso proxecto realizouse unha análise de requisitos vistos con anterioridade no apartado 5. Unha vez definido o obxectivo do proxecto, procedeuse a empezar co traballo fixando obxectivos cada dúas semanas acordando reunións entre o cliente (titor académico) e o alumno. A medida que se cumpren ou non os obxectivos, marcamos outros ou valoramos un posible plan de continxencia.

Capítulo 7

Planificación e custos

A organización das tareas, a súa estrutura e o manexo dos tempos é esencial nos proxectos. Neste capítulo falaremos dos recursos dos que dispoñemos, tanto a nivel tanxible como intanxible, e como organizamos o traballo para ter un bo producto o rematar o proxecto.

7.1 Recursos

Nesta sección falarase sobre os distintos medios ou axudas que se empregaron para conseguir a finalidade do proxecto.

7.1.1 Recursos materiais

Os recursos materiais necesarios para este proxecto son:

- Raspberry Pi 3 y Raspberry Pi Zero W. Son os dispositivos que funcionarán como rastrexadores BLE e como centro de procesado dos datos.
- Balizas BLE. Son os dispositivos BLE a escanear. Poden ser beacons, smartphones, ou calquer dispositivo con BLE.
- Ordenador portátil para o desenvolvemento do proxecto con sistema operativo Linux. Poderíase traballar sobre as propias Raspberry, pero por comodidade é mellor traballar sobre un portátil.
- Baterías externas de 5000 mAh ou 10000 mAh. No caso de que non podamos conectar as Raspberry a unha toma de corrente é necesario provelas dunha fonte de enerxía.
- Tarxetas SD de 16 Gb ou 32 Gb. As Raspberry necesitan ter unha imaxe do sistema operativo, esta imaxe debe estar neste tipo de tarxetas.

7.1.2 Recursos software

Os recursos software empregados no sistema, e que explicamos e detallamos os motivos da súa elección na sección 4, son os seguintes:

- Sistema operativo Raspbian.
- Librería *Bluepy*.
- Base de datos SQlite3.
- Entorno de desenvolvemento Django, Pycharm.
- Utilidade *batctl*.
- MQTT e *broker* MQTT Mosquitto.
- Librería *Localization*.
- Librerías para cálculos matemáticos: *numpy*, *scipy*, etc.
- Entorno de desenvolvemento para Vue.js, Visual Studio Code.
- Google Maps.

7.1.3 Recursos humanos

- Cliente. O cliente establece cales son os requisitos e encárgase de validar as funcionalidades.
- Analista. Encárgase de analizar os requisitos do cliente e convertilo en funcionalidades a implementar.
- Programador de BackEnd. Encárgase de implementar a lóxica de negocio e expoñela mediante unha aplicación.
- Deseñador. Encárgase de crear os *mockups* da interface.
- Maquetador de FrontEnd. Encárgase de implementar o deseño da interface e conectala coa aplicación.
- Experto. Encárgase de proporcionar asesoramento no proxecto.
- Xefe do proxecto. Encárgase de supervisar o proxecto e contactar co cliente.

Táboa 7.1: Responsabilidades dos diferentes roles implicados no proxecto.

Rol	Persoa encargada
Cliente	Director do proxecto (Tutor)
Analista	Alumno
Programador BackEnd	Alumno
Deseñador	Alumno
Maquetador de FrontEnd	Alumno
Experto	Director do proxecto
Xefe do proxecto	Alumno

Responsabilidades

Na táboa 7.1 pódemos consultar que persoas asumirán cada rol.

7.2 Planificación

7.2.1 Tempo dos recursos humanos que dispoñemos

Para unha boa planifación debemos ter unha aproximación do tempo do que dispoñemos, polo que sabendo que empezamos o proxecto o 01/09/2021 e a data de entrega é o 23/02/2021 podemos estimar que teremos a posibilidade de facer case 13 *sprints*, supoñendo que cada un dura de media dúas semanas. Na realidade os *sprints* varían en tempo, debido a que algunhas tarefas levarán máis do esperado e outras menos. Na táboa 7.2 podemos ver a estimación da que falamos.

Outro factor a ter en conta é que a nosa dispoñibilidade horaria estará limitada as tardes de diario e aos fins de semana debido a ter un emprego durante a realización deste proxecto.

7.2.2 Aproximación inicial

Tendo en conta que dispoñemos de 13 *sprints* (12 de dúas semanas e 1 dunha semana) faise unha primeira valoración dos obxectivos necesarios é procedemos a marcar uns obxectivos iniciais dos *sprints* que logo varían segundo as necesidades do proxecto. Estes son os obxectivos marcados nunha primeira aproximación para cada *sprint* e que veremos o seu seguemento na sección 7.2.3:

- Sprint 1. Análise de requisitos, estudo de traballos previos e deseño da arquitectura a seguir.
- Sprint 2. Preparación dos dispositivos BLE e os rastrexadores.
- Sprint 3. Creación da rede de rastrexadores e script de lectura BLE.

Táboa 7.2: Sprints esperados.

Nombre sprint	Inicio	Fin
Sprint 1	01/09/2020	14/09/2020
Sprint 2	15/09/2020	28/09/2020
Sprint 3	29/09/2020	12/10/2020
Sprint 4	13/10/2020	26/10/2020
Sprint 5	27/10/2020	09/11/2020
Sprint 6	10/11/2020	23/11/2020
Sprint 7	24/11/2020	07/12/2020
Sprint 8	08/12/2020	21/12/2020
Sprint 9	22/12/2020	05/01/2021
Sprint 10	06/01/2021	18/01/2021
Sprint 11	19/01/2021	01/02/2021
Sprint 12	02/02/2021	15/02/2021
Sprint 13	15/02/2021	20/02/2021

- Sprint 4. Creación do BackEnd e modelo de datos.
- Sprint 5. Traballo de BackEnd. Creación dos EndPoints.
- Sprint 6. Conexión script de lectura-BackEnd. Primeiras probas.
- Sprint 7. Conexión script-BackEnd mediante MQTT.
- Sprint 8. Creación da interface. Primeiras visualizacións.
- Sprint 9. Creación do cálculo da posición en segundo plano.
- Sprint 10. Realización de probas do sistema.
- Sprint 11. Posibles melloras no sistema.
- Sprint 12. Documentación da memoria.
- Sprint 13. *Sprint* de continxencia.

7.2.3 Seguemento do traballo

- Sprint 1. O primeiro *sprint* foi completado con éxito no tempo establecido. Nel depuráronse os requisitos aos que facemos referencia no capítulo 5 e optouse pola arquitectura descrita no capítulo 8. Ademais realizouse traballo de investigación revisando o estado do arte.

- Sprint 2. No segundo *sprint* realizouse a preparación dos dispositivos, instalando o sistema operativo Raspbian en cada un dos rastrexadores, e configurando os parámetros de rede para poder traballar en remoto.
- Sprint 3. Neste *sprint* traballouse en crear a rede de rastrexadores en forma de malla. Primeiro fíxose un traballo de investigación para ver como poder instalar a rede nas Raspberry o que consumiu gran parte do *sprint* e tivo que ampliarse o tempo do *sprint* unha semana na que terminouse de crear a rede en forma de malla das Raspberry. Tamén creouse o script de lectura, ademais de probar que efectivamente lía os dispositivos.
- Sprint 4. Nesta etapa creouse un proxecto Django para o BackEnd e ademais creouse o modelo de datos. O *sprint* foi completado en plazo.
- Sprint 5. Para este *sprint* buscouse como obxectivo crear os EndPoints para poder expoñelos na API REST e así poder ser accesibles dende a interface. A igual que o anterior completouse en plazo.
- Sprint 6. Para probar a integración do *script* co proxecto Django enlazouse mediante a API REST as dúas partes e probouse a realizar os primeiros cálculos de posicións. O *sprint* tamén foi terminado en plazo.
- Sprint 7. Unha vez probado que o sistema era capaz de procesar os datos escaneados, decidiuse cambiar o enlazado empregando o protocolo MQTT para realizar unha conexión máis lixeira entre as dúas partes. O enlazado levou menos tempo do esperado e acabouse o *sprint* unha semana antes do esperado.
- Sprint 8. A estas alturas cun sistema medianamente funcional procedeuse a crear o proxecto do FrontEnd para ter unha interface de usuario coa que se puidese localizar visualmente os dispositivos mediante Google Maps e tamén configurar os parámetros do sistema. O *sprint* levou máis tempo do esperado é tivo que retrasarse unha semana.
- Sprint 9. Tras as primeiras visualizacións do sistema detectouse que o cálculo das posicións debe realizarse en segundo plano se queremos obter posicións históricas das balizas. Para isto creouse o servizo de localización en segundo plano que cos datos dos escaneamentos realizaba o cálculo da posición. Ademais tívose que restruturar o modelo para incluir esta nova entidade.
- Sprint 10. Neste *sprint* realizáronse as probas da independecia da rede, probas de calibración, probas de localización de balizas estáticas e en movemento. Tamén se realizou a análise dos datos acadados. O *sprint* levou máis tempo do esperado e retrasouse unha semana.

Táboa 7.3: Custos materiais do proxecto.

Recurso	Coste(€)	Imputado(€)
Ordenador portátil HP	650	0
Raspberry Pi 3 (3 Uds)	150	150
Raspberry Zero W (3 Uds)	55	55
Batería externa 10000 mAh (3 Uds)	45	45
Cable alimentación Micro USB B (3 Uds)	20	20
Balizas BLE IBKS 105 (4 Uds)	70	70
Google Maps Javascript	0	0
Total	**990€**	**340€**

- Sprint 11. Unha vez probado o sistema analizouse cales eran as melloras que podían facerse no sistema e optouse por mellorar o aspecto da interface e implementar a conexión mediante un Bridge a un dos nodos.
- Sprint 12. Aínda que durante todo o proxecto fóronse documentando todos os pasos, este *sprint* dedicouse a terminar esa documentación e preparar a memoria do traballo.
- Spritn 13. Este era o *sprint* de continxencia por se ocorría algun problema ter marxe para poder rematar o proxecto en plazo. Como ao final produciuse algún retraso nalgún dos *sprint* este tempo foi consumido.

7.3 Custos

Empregando como base os datos da planificación final do proxecto e os recursos (materiais e humanos) dispoñibles, estimouse o custo total do proxecto.

7.3.1 Custos materiais e software

Para detallar os gastos nos recursos materiais creouse a táboa 7.3 que amosa o investimento necesario para ter dispoñibles estes recursos. Algúns dos recursos non teñen un prezo imputado ou estar dispoñibles ou ser unha versión gratuita como é o caso de Google Maps.

7.3.2 Custos dos recursos humanos

Os recursos humanos, sen ningunha dúbida, representan o activo máis importante dentro da cadea de valor e é importante estimar a utilización dos nosos recursos. No caso específico dos recursos humanos dáse unha dualidade a razón de que como tal é un recurso, pero ao mesmo tempo, son persoas, por tanto debe existir un balance que permita extraer o mellor deles sen entrar na explotación. Tendo en conta o tempo estimado para o desenvolvemento

Táboa 7.4: Custos estimados dos recursos humanos do proxecto.

Recurso	**Salario (€/hora)**	**T. estimado(hora)**	**C. estimado(€)**
Analista	35	75	2.625
Prog. BackEnd	25	150	3.750
Deseñador	40	25	1.000
Maq. de FrontEnd	22	150	3.300
Experto	50	25	1.250
Xefe do proxecto	50	40	2.000
Total	-	-	**13.900€**

Táboa 7.5: Custos reais dos recursos humanos do proxecto.

Recurso	**Salario (€/hora)**	**T. estimado(hora)**	**C. estimado(€)**
Analista	35	80	2.800
Prog. BackEnd	25	160	4.000
Deseñador	40	25	1.000
Maq. de FrontEnd	22	120	2.640
Experto	50	28	1.400
Xefe do proxecto	50	45	2.250
Total	-	-	**14.990€**

do proxecto na planificación inicial, o custo dos recursos humanos para o noso proxecto é o que se detalla na táboa 7.4.

Como toda estimación, na realidade este cálculo tende a variar. Para ver a tempo real dos recursos humanos empregados podemos consultar a táboa 7.5.

Capítulo 8

Arquitectura, modelo de datos e interface

Neste capítulo exponse as decisións estruturais do sistema, a topoloxía, as decisións técnicas empregadas para a creación da aplicación, o modelo de xestión de datos, a accesibilidade a eles, e as funcionalidades expostas ao usuario mediante a interface deseñada.

8.1 Arquitectura a nivel físico

Nesta sección falaremos de como se desenvolveu o proxecto dende un punto de vista dos dispositivos a nivel hardware, o proceso para poder crear unha comunicación entre eles e como os situaríamos nun posible despregamento nun entorno.

8.1.1 Rede

A arquitectura a nivel hardware seguida para este proxecto está baseada no concepto de rede de malla tal como comentábase anteriormente na sección 4.4. Para lembralo, debemos ter en conta que cunha rede de malla non hai ningún punto de acceso central.

A rede estará formada por dispositivos (no noso caso os rastrexadores) que realizan conexións directas entre eles cando estes dispositivos estean ao seu alcance (veciños). O propio protocolo de rede de malla e os algoritmos que xestionan a rede, son os encargados de enrutar os paquetes de datos. Deste xeito calquera dispositivo que forme parte da rede pode comunicarse con tódolos demais sempre que haxa unha ruta dispoñible.

No noso caso contamos cunha serie de Raspberry, coas que podemos formar unha rede deste tipo e facer comunicacións entre os dispositivos sen a necesidade de empregar un router ou similar. A rede pode ser tan grande como desexemos, pero debemos ter en conta que a calidade da conexión ven limitada polos camiños entre os dispositivos, polo que se entre dous

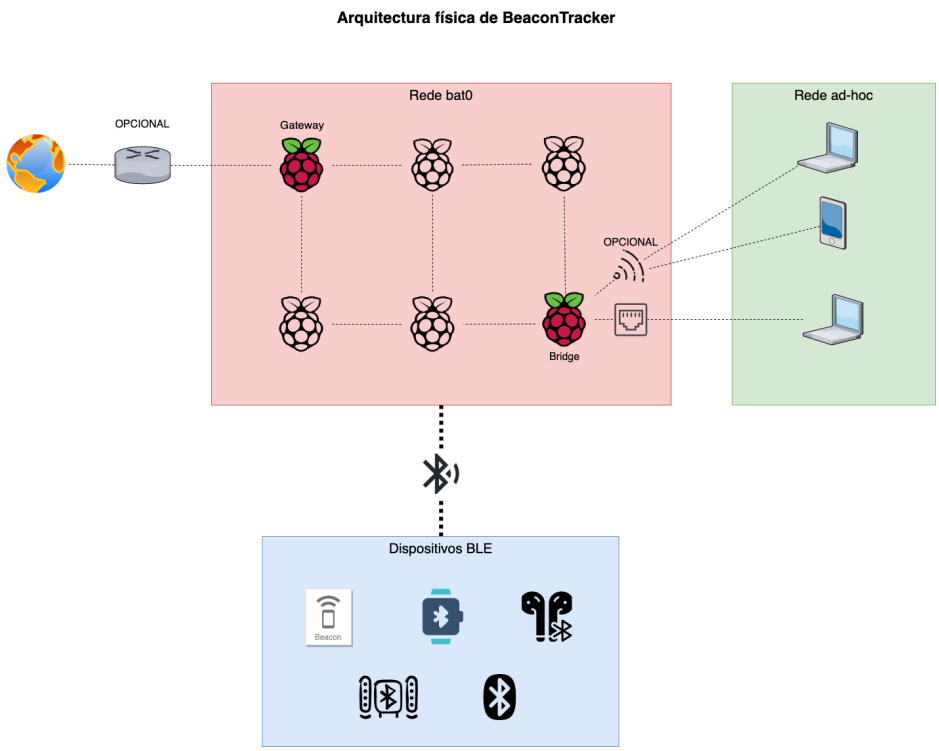


Figura 8.1: Arquitectura a nivel físico do noso sistema de localización en interiores.

dispositivos só existe un camiño a conexión pode verse afectada se a distancia é moi grande ou un destes dispositivos intermedios cae.

Con este tipo de rede tamén temos a posibilidade de establecer un ou varios nodos como Gateway cos que nos poderemos conectar a outras redes externas como por exemplo Internet. Esta utilidade nos permite ter máis recursos para a nosa aplicación, a pesares de que non sexan necesarios para o noso sistema. Outra cousa que podemos configurar é establecer un ou varios dos nodos como Bridge o que nos permitiría conectar dispositivos á rede do sistema de localización co obxectivo de empregar as funcionalidades do sistema.

Con esta idea o que conseguimos é que cada rastrexador poda escanear o seu entorno e detectar os cambios na forza do sinal dos dispositivos BLE e a través da rede poder recollelos e procesalos para poder estimar a posición de cada un deles.

Na imaxe 8.1 podemos ver o concepto seguido para a nosa arquitectura.

8.1.2 Visión espacial

Para poder ver de xeito máis visual como sería a infraestrutura montada nun posible entorno, na imaxe 8.2 vemos a disposición dos rastrexadores situados en diferentes habitacións. Esta colocación dos rastrexadores formaría unha rede entre eles (liñas verdes) que permitiría

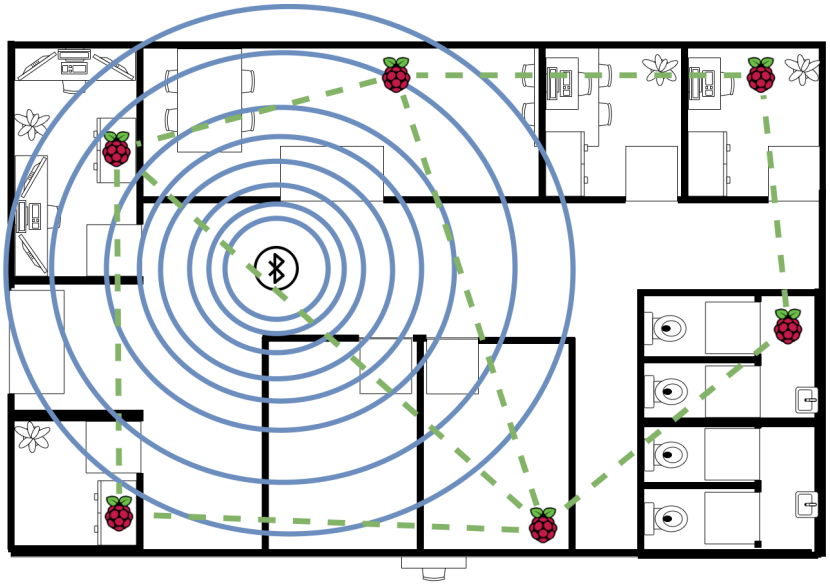


Figura 8.2: Posible situación dos dispositivos rastrexadores nun edificio típico.

que cada un deles detecte calquer baliza que estea ubicada nas inmediacións do andar para posteriormente procesar os datos escaneados.

8.2 Arquitectura a nivel lóxico

Para implementar o sistema de localización deste proxecto a nivel de software, optouse por unha estrutura que separa a parte da interface de usuario (FrontEnd) da parte de xestión dos datos (BackEnd). O beneficio desta arquitectura é a modularidade que presenta separar a interface da lóxica interna. Esta modularidade permite modificar ou desenvolver calquera das partes de xeito independente coa vantaxe de empregar diferentes tecnoloxías.

O BackEnd, desenvolto en Django 4.1.1, xestiona os datos recollidos por un proceso que se executa en segundo plano e que se encarga de escanear de forma periódica as variacións do RSSI. Para este proceso de recollida, debemos situar o BackEnd nun dos dispositivos para centralizar os datos na base de datos e despois poder cálcular as posicións.

Os motivos de usar unha base de datos centralizada é a complexidade da xestión dunha base de datos distribuida a través dunha rede de malla. A complexidade xorde das desvantaxes que ten este tipo de base de datos como son: a falta de estándares no desenvolvemento e implementación, problemas de rendemento e fiabilidade debido a esta falta de estándares e sobre todo, porque usar este tipo de base de datos implicaría un incremento exponencial no tempo de desenvolmento o que se traduce nun maior impacto económico [38].

Entonces, Onde situamos o BackEnd? O sistema o situaremos no nodo de punto de acceso (Gateway) por varios motivos: por simplicidade a nivel de enrutado, este dispositivo

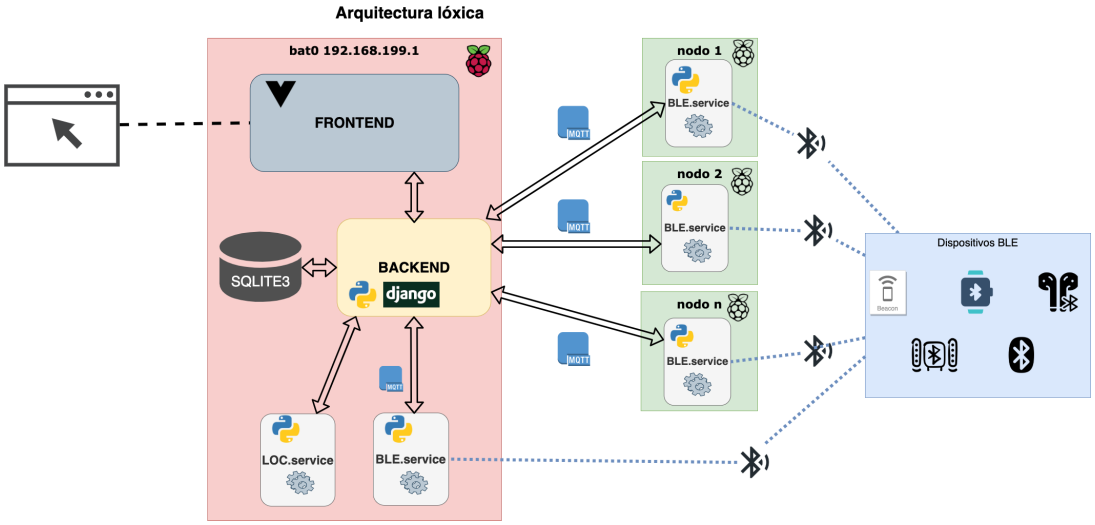


Figura 8.3: Proposta de arquitectura a nivel lóxico do sistema de localización en interiores.

establécese como unha IP estática; por ser o nodo con acceso a redes externas, polo que terá a maior calidade de conexión en caso de que usemos esta rede; e por motivos de facilitar a accesibilidade.

Antes falabamos de que o BackEnd recollía os datos dun servizo que escaneaba o RSSI periodicamente. Este servizo, implementado como un script Python, estará situado en cada un dos rastrexadores para abarcar o maior espazo posible co menor número de dispositivos. Para comunicar este escaneamento co BackEnd empregaremos o protocolo MQTT (ver Sec.4.4) que mediante a publicación nun *topic*, ao que o BackEnd estará suscrito e será notificado no momento de cada lectura, poderemos comunicar os escaneamentos.

Por outra parte, o cálculo das posicións das balizas deben tamén calcularse de xeito periódico, polo que outro proceso en segundo plano debe recoller os datos dos escaneamentos e facer os cálculos e incorpóralos á base de datos que xestiona o BackEnd. Para este proceso optouse por outro script Python que neste caso situaremos no mesmo dispositivo co BackEnd, aínda que se quixeramos escalar, poderiamos repartir os cálculos entre varios nodos. Nesta implamentación non é necesario.

Por último a interface (FrontEnd), implementada en Vue.js 4.4.1, será o medio de comunicación entre o usuario e o sistema, aparte da API REST do BackEnd. Esta interface tamén a situaremos no mesmo dispositivo co BackEnd polo mesmo motivo, a sinxeleza para a accesibilidade dos usuario.

8.3 Modelo de datos | Entidade-relación

Como case toda aplicación, o noso sistema necesita almacenar, procesar, ou consultar toda a posible información recollida nos diferentes procesos executados. Para isto necesitamos crear un modelo de datos. Un modelo de base de datos é un tipo de modelo de datos que determina a estrutura lóxica dunha base de datos e de maneira fundamental determina o modo de almacenar, organizar e manipular os datos nesa base. O modelo da base de datos nas bases relacionais adoita ser independente da tecnoloxía empregada e por iso é importante definir como gardaremos a información. Para a definición do modelo de base de datos adóitase usar o modelo entidade-relación [39]. Na imaxe 8.4 podemos ver o diagrama entidade-relación creado para o noso modelo de datos e que expón o seguinte:

- **Baliza**. Esta entidade corresponde aos dispositivos que se van a localizar. Nesta táboa se garda información importante para o escaneamento, como a mac, e importante para o cálculo da posición como a potencia media de emisión.
- **Rastreador**. Para un rastrexador debemos gardar a súa mac para identificalo dos demais e ademais deberá estar situado nunha localización.
- **Escaneo**. Un escaneamento é feito por un rastrexador e está asociado a unha baliza cun RSSI determinado nun data concreta.
- **Localizacion**. Corresponde as localizacións que pode ter un rastrexador, para eso deberemos saber a súa latitude e lonxitude.
- **LocalizacionRastreador**. Un rastrexador pode cambiar a súa localización durante o tempo polo que é importante establecer un histórico das súas localizacións.
- **Posicion**. Corresponde ás posicións calculadas para as balizas, polo que unha baliza estará nunha posición durante un intervalo de tempo calculado.

8.4 API

Para interactuar co sistema para obter os datos ou executar algunhas das funcións implememtadas, empregamos a API do BackEnd que permite comunicar o que se desexa ao sistema, para que este comprenda a solicitude e a cumpra. Podemos considerar a API como o mediador entre os usuarios ou clientes e os recursos que queren obter.

No noso caso esta API está integrada no BackEnd e mediante solicitudes a través de HTTP podemos mandar ou recibir respostas co formato JSON. As funcionalidades expostas para a comunicación son as seguintes:

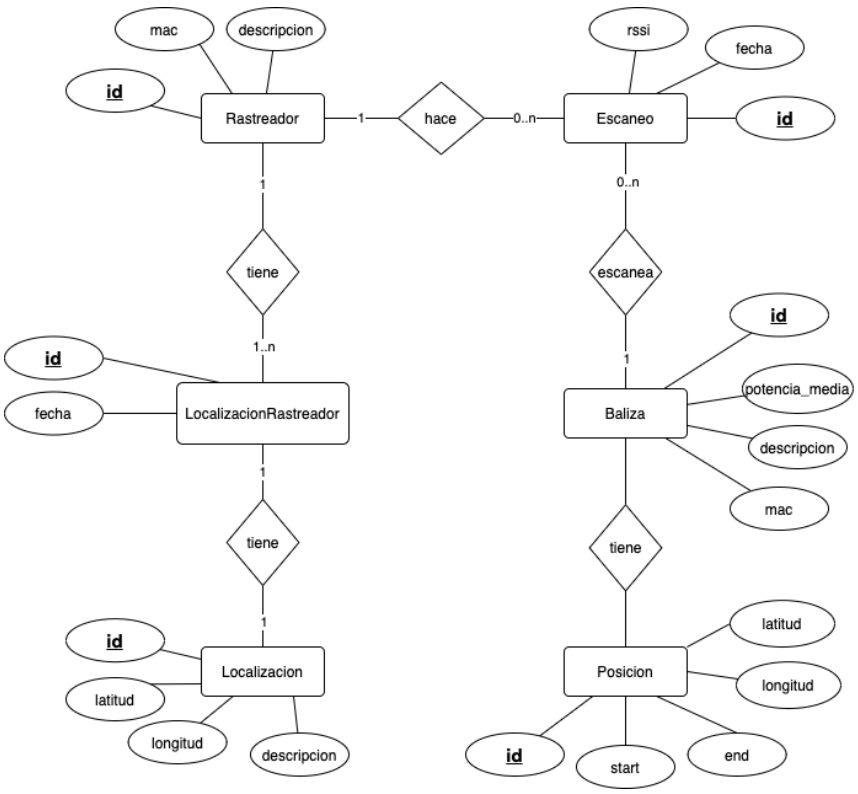


Figura 8.4: Diagrama entidade-relación da base de datos integrada no BackEnd do sistema de localización.

- EndPoints para balizas (Táboa 8.1): Estes servizos permiten a xestión da entidade **Baliza** do modelo da base de datos. Ademais da xestión na base de datos, notifican os rastrexadores, mediante un topic MQTT, os cambios publicados neste modelo para que podan realizar os escaneamentos correctamente.

- EndPoints para rastrexadores (Táboa 8.2): Estes servizos permiten a xestión da entidade **Rastreador** do modelo da base de datos. Ademais da xestión na base de datos, permite que a información dos escaneamentos dos rastrexadores sexa recibida polo BackEnd, e mediante un topic MQTT, notifícalle na súa creación, as balizas que debe escanear.

- EndPoints para localizacións (Táboa 8.3): Estes servizos permiten a xestión da entidade **Localizacion** do modelo da base de datos. Ademais da xestión na base de datos, permite a localización histórica dos rastrexadores (entidade **LocalizacionRastreador**) así como asociar unha nova localización a un rastrexador para cambialo de ubicación.

- EndPoints para escaneamentos (Táboa 8.4): Estes servizos permiten a creación da entidade **Escaneo** do modelo da base de datos. Ademais da creación na base de datos, permite recoller o escaneamentos para unha baliza nun intervalo de tempo dado, así

Táboa 8.1: Servizos proporcionados pola API do BackEnd para as balizas (EndPoints).

Método	Url	Descripción
GET	/api/beacons/	Lista todas as balizas cos atributos de identificador, descripción, mac e a potencia media.
GET	/api/beacons/<id>/	Devolve os mesmos atributos que na lista, pero para unha baliza en concreto.
POST	/api/beacons/	Crea unha baliza, e ademais notifica a creación mediante MQTT para que os rastrexadores actualicen que balizas deben escanear.
PUT	/api/beacons/<id>/	Actualiza os parámetros da baliza.
DELETE	/api/beacons/<id>/	Borra unha baliza e comunícallo os rastrexadores.

Táboa 8.2: Servizos proporcionados pola API do BackEnd para rastrexadores (Endpoints).

Método	Url	Descripción
GET	/api/devices/	Lista todos os rastrexadores cos atributos de identificador, descripción, mac e última localización rexistrada para ese rastrexador.
GET	/api/devices/<id>	Devolve os mesmos atributos que na lista pero para un rastrexador en concreto.
POST	/api/devices/	Crea un rastrexador, e ademais notifica a creación mediante MQTT para que o rastrexador poida recibir a configuración de escanneo.
PUT	/api/devices/<id>/	Actualiza os parámetros do rastrexador.
DELETE	/api/devices/<id>/	Elimina un rastrexador.

como o seu RSSI medio, a súa varianza e a súa conversión a distancia en metros, calculado co RSSI.

- EndPoints para as posicións calculadas das balizas. (Táboa 8.5): Estes servizos permiten a xestión da entidade **Posicion** do modelo da base de datos. Esta entidade **Posicion** é calculada por un proceso en segundo plano e é accesible mediante estes servizos. Ademais permite recibir a posición media durante un intervalo de tempo tanto en formato JSON como en formato CSV.

- EndPoints para configuración do sistema (Táboa 8.6): Estes servizos permiten ver e modificar os parámetros do sistema (modo de resolución da multilateración, factor ambiental, etc.).

Táboa 8.3: Servizos proporcionados pola API do BackEnd para as localizaciones (Endpoints).

Método	Url	Descripción
GET	/api/locations/	Obtén todas as localizacións.
GET	/api/locations/<id>/	Obtén a localización indicada.
POST	/api/locations/	Crea unha localización.
GET	/api/locations/<id>/tracker	Devolve o histórico de localizacións para un rastrexador.
POST	/api/locations/asociate/	Asocia unha localización a un rastrexador.

Táboa 8.4: Servizos proporcionados pola API do BackEnd para os escaneamentos (Endpoints).

Método	Url	Descripción
POST	/api/scans/	Crea un escaneamento
GET	/api/scans/?start=<time>&end=<time> &mac=<mac>	Obtén os escaneamentos para unha baliza no intervalo dado.

8.5 Interface

A interface de usuario é o medio co que o usuario pode comunicarse co sistema de xeito amigable. O obxectivo da interface é dar unha conexión entre o sistema e o usuario de xeito que sexa fácil de entender, fácil de accionar e intuitiva. Para decidir que funcionalidades se deben expoñer, realizamos un deseño mediante wireframes nos que decidimos as utilidades da nosa interface. Na imaxe 8.5 podemos ver este deseño.

Deste xeito decidimos que a interface debe poder xestionar as balizas, os rastrexadores, ver as balizas en tempo real, o seu movemento durante un intervalo de tempo e poder configurar os parámetos. Para ver como sería o resultado na sección 9.5 do capítulo de implementación falaremos do resultado.

Táboa 8.5: Servizos proporcionados pola API do BackEnd para as posicións calculadas (End-points).

Método	Url	Descripción
POST	/api/position/	Crea unha posición para unha baliza dada.
GET	/api/position/	Obtén un listado das posicións en tempo real de todas as balizas.
GET	/api/position/<idBaliza> ?start=<Datetime>&end=<Datetime> &frequency=<frecuencia>	Obtén as posicións medias durante un intervalo de tempo e cunha frecuencia dada.
GET	/api/position/csv/	Obtén un csv das posicións en tempo real de todas as balizas.
GET	/api/position/<idBaliza>/csv/ ?start=<Datetime>& end=<Datetime> &frequency=<frecuencia>	Obtén un csv das posicións medias durante un intervalo de tempo e cunha frecuencia dada.
GET	/api/position/<idBaliza>/csv_scan/ ?start=<Datetime>&end=<Datetime> &frequency=<frecuencia>	Obtén datos sobre a análise dos escaneamentos, como a varianza, rssi medio, etc.

Táboa 8.6: Servizos proporcionados pola API do BackEnd para a configuración.

Método	Url	Descripción
GET	/api/configuration/	Obtén os parámetros de configuración do sistema.
PUT	/api/configuration/change/	Modifica os parámetros de configuración do sistema.

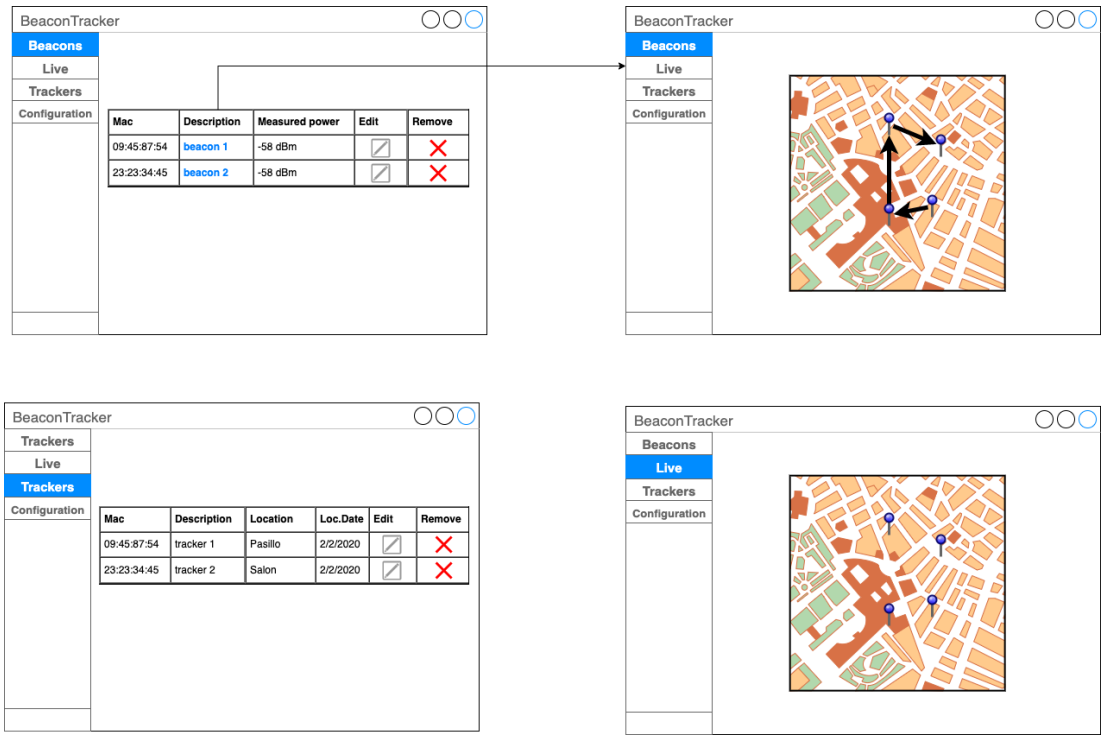


Figura 8.5: *Wireframes* da interface de usario.

Capítulo 9

Implementación

Neste capítulo recóllense os detalles da implementación realizada para cada unha das partes do sistema: a base de datos, os rastrexadores, o BackEnd, os servizos, etc. Ademais coméntanse os principais aspectos da interface gráfica de usuario.

9.1 Rede

9.1.1 Configuración da rede en malla

Para crear a rede de malla na Raspberry Pi débese usar *batman-adv*, que é parte do kernel estándar de Linux. Para isto, imos configurar o módulo do kernel *batman-adv* para tomar o control da interface Wi-Fi *wlan0* e crear unha rede malla sobre Wi-Fi. *Batman-adv* creará unha nova interface *bat0* para permitir que os rastrexador envíe tráfico de rede a través da rede en malla. Para establecer esta nova interface *bat0* debemos seguir os seguintes pasos en todas as Raspberry que queremos que formen parte da rede de malla, incluídos os nodos de porta de enlace (Gateway) e o ponte (Bridge):

1. Para administrar a rede de malla, é necesario instalar unha utilidade chamada *batctl* . Isto pódese facer usando o comando **sudo apt-get install -y batctl**.

2. Agora débese crear o arquivo */start-batman-adv.sh* co seguinte contido:

```
#!/bin/bash
# batman-adv interface to use
sudo batctl if add wlan0
sudo ifconfig bat0 mtu 1468
# Tell batman-adv this is a gateway client
sudo batctl gw_mode client
# Activates batman-adv interfaces
sudo ifconfig wlan0 up
sudo ifconfig bat0 up
```

3. O arquivo *start-batman-adv.sh* debe ser executable, para iso hai que empregar o comando **chmod +x /start-batman-adv.sh**.

4. O seguinte paso é crear a definición de interface de rede para a interface *wlan0* no arquivo */etc/network/interfaces.d/wlan0*

```
auto wlan0
iface wlan0 inet manual
    wireless-channel 1
    wireless-essid red-tfg
    wireless-mode ad-hoc
```

5. Agora hai que asegurarse de que o módulo do kernel *batman-adv* estea cargado no momento do arranque emitindo o comando

   echo 'batman-adv' | sudo tee –append /etc/modules

   e deter o proceso DHCP para que non intente administrar a interface LAN inalámbrica co comando

   echo 'denyinterfaces wlan0' | sudo tee –append /etc/dhcpcd.conf

6. Por último hai que asegurar que se chama ao script de inicio e para iso editamos o arquivo */etc/rc.local* como usuario root e engadimos o seguinte:

```
/home/pi/start-batman-adv.sh &
```

9.1.2 Configuración da porta de enlace

Despois de ter todos os nodos creados hai que definir unha porta de enlace e unha ponte. Para a porta de enlace debemos escoller unha das Raspberry, dado que esta porta utiliza o encamiñamento IP para permitir que o tráfico pase de forma selectiva entre a malla e as outras redes. A rede en malla debe ter un rango de direccións diferente. Este rastrexador será o servidor DHCP para a rede en malla. DHCP é o servizo que proporciona a configuración de rede aos dispositivos. Para realizar esta configuración como nodo de porta de enlace debemos facer o seguinte:

1. Instalar o software DHCP co comando: **sudo apt-get install -e dnsmasq**

2. Configurar o servidor DHCP editando o arquivo *dnsmasq.conf* como usuario root e engadir as seguintes liñas:

```
interface=bat0
dhcp-range=192.168.199.2,192.168.199.99,255.255.255.0,12h
```

3. Por último, hai que cambiar o arquivo de inicio para agregar as regras de encamiñamento para reenviar o tráfico de malla á outra rede e realizar a tradución de direccións de rede na resposta, configurar o nodo como unha porta de enlace e tamén configurar a dirección IP da interface desta porta. Para facer isto, hai que actualizar o arquivo *start-batman-adv.sh* e cambiar o contido a:

```
#!/bin/bash
# batman-adv interface to use
sudo batctl if add wlan0
sudo ifconfig bat0 mtu 1468
# Tell batman-adv this is an internet gateway
sudo batctl gw_mode server
# Enable port forwarding
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o bat0 -m conntrack --ctstate
    RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i bat0 -o eth0 -j ACCEPT
# Activates batman-adv interfaces
sudo ifconfig wlan0 up
sudo ifconfig bat0 up
sudo ifconfig bat0 192.168.199.1/24
```

9.1.3 Configuración do nodo ponte

Por outra parte quedaría o nodo ponte. Un nodo ponte permite que os dispositivos sen malla usen os nodos en malla. O nodo de porta de enlace proporciona o servidor DHCP que tamén servirá aos dispositivos con ponte, xa que as solicitudes de DHCP flúen a través dunha ponte.

Para configurar un nodo como ponte debemos facer o seguinte:

1. Instalar as utilidades da ponte usando o comando **sudo apt-get install -e bridge-utils**

2. Crear unha configuración de interface para a interface *eth0*. Isto permitirá que o porto ethernet conéctese en quente, o que significa que o cable ethernet pódese conectar e desconectar. Para facer isto hai que crear o arquivo */etc/network/interfaces.d/eth0* como usuario root e incluir o seguinte:

```
auto eth0
allow-hotplug eth0
iface eth0 inet manual
```

3. Modificar o arquivo */etc/dhcpcd.conf* como root e engadir na ultima liña o seguinte:

```
denyinterfaces wlan0 eth0 bat0
```

4. Por último debemos editar o arquivo */start-batman-adv.sh* para configurar a ponte:

```
#!/bin/bash
# batman-adv interface to use
sudo batctl if add wlan0
sudo ifconfig bat0 mtu 1468
sudo brctl addbr br0
sudo brctl addif br0 eth0 bat0
# Tell batman-adv this is a gateway client
sudo batctl gw_mode client
# Activates batman-adv interfaces
sudo ifconfig wlan0 up
sudo ifconfig bat0 up
# Restart DHCP now bridge and mesh network are up
sudo dhclient -r br0
sudo dhclient br0
```

9.2 BackEnd

O BackEnd é a parte encargada de que toda a lóxica do sistema funcione. Consiste no conxunto de accións que pasan dentro do sistema pero que non podemos ver. Algunhas das accións que controla o BackEnd son a conexión coa base de datos ou a comunicación co servidor. Na imaxe 9.1 podemos ver o esquema seguido para estruturar o software do BackEnd no que temos podemos destacar o patrón seguido e implementado.

9.2.1 Patrón seguido para o BackEnd

Django é un framework MTV (unha modificación de MVC, ver Fig. 9.2 [40]). Isto débese a que os programadores non tiveron a intención de seguir algún patron de desenvolvemento, senón facer o framework o máis funcional posible. Para entender o patrón MTV debemos fixarnos na analogía con MVC.

- O Modelo en Django segue sendo o Modelo. O modelo define os datos almacenados, atópase en forma de clases de Python, cada tipo de dato que debe ser almacenado atópase nunha variable con certos parámetros, posúe métodos tamén. Todo isto permite indicar e controlar o comportamento dos datos.

- A Vista en Django chámase Plantilla (*Template*). A Plantilla é basicamente unha páxina HTML con algunhas etiquetas extras propias de Django. A Plantilla recibe os datos da vista e logo organízaos para a presentación ao navegador web. No noso caso particular desacoplamos esta parte de Django. A plantilla será o modulo FrontEnd e farase en Vue.js, polo que no noso proxecto de Django non teremos esta parte deste patrón. Isto permite que a lóxica do sistema siga permanecendo na vista.

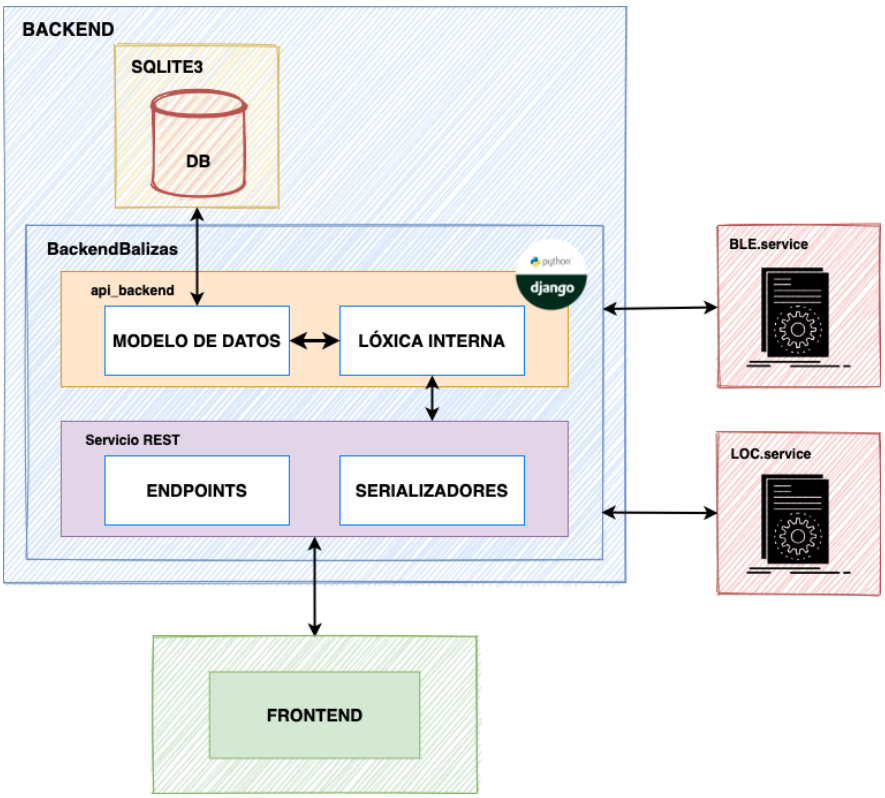


Figura 9.1: Esquema do software do BackEnd.

Figura 9.2: Modelo MTV de Django.

- O Controlador en Django chámase Vista. A Vista preséntase en forma de funcións en Python, o seu propósito é determinar que datos serán visualizados. Django permite escribir código Python en lugar de SQL para facer as consultas que necesita a Vista. O máis importante a entender con respecto á Vista é que non ten nada que ver co estilo de presentación dos datos, só se encarga dos datos. A presentación é tarefa do persoal.

9.2.2 Implantación no noso proxecto

Como se comentaba anteriormente, no noso proxecto Django só estará presente a parte do Modelo e da Vista, polo que se vemos de novo a imaxe 9.1 podemos ver como están estruturadas e podemos comentar o seguinte sobre elas:

- Modelo: As aplicacións web de Django acceden e administran os datos a través de obxectos de Python aos que se fai referencia como modelos. Os modelos definen a estrutura dos datos almacenados, incluídos os tipos de campo e os atributos de cada campo, como o seu tamaño máximo, valores predeterminados, lista de selección de opcións, texto de axuda para a documentación, texto de etiqueta para formularios, etc. A definición do modelo é independente da base de datos subxacente. Unha vez que se elixe a base de datos, non necesitamos falar directamente con ela. Simplemente escribimos a estrutura do modelo e algo de código, e Django encargase de todo o traballo sucio, ao comunicarse coa base de datos por nós. En Django os modelos están definidos, normalmente, no arquivo models.py da aplicación. No seguinte código podemos ver un exemplo para a entidade Baliza.

```
class Baliza(models.Model):
    mac = models.CharField(unique=True, null=False, max_length=100)
    descripcion = models.CharField(null=True, max_length=100)
    potencia_media = models.FloatField(null=False)

    def __str__(self):
        return "Baliza:" + str(self.mac) + " con " + str(self.potencia_media)

    def crear_baliza(data):
        try:
            baliza = Baliza.objects.create(mac=data['mac'].upper() \
            ,descripcion=data['descripcion'] \
            .upper(),potencia_media=data['potencia_media'])
            return baliza
        except IntegrityError:
            return None
```

- Vista: Antes mencionamos que a Vista se corresponde co Controlador do patrón MVC, polo que no noso caso isto englobará os EndPoints, os serializadores dos datos e a lóxica de negocio. Estas funcionalidades son as rutinas que realizan entradas ou consultas aos datos, xeración de informes e máis especificamente todo o procesamento que se realiza detrás da aplicación visible para o usuario. No seguinte código podemos observar un exemplo de como controlamos a información do sistema mediante estas utilidades.

```
class BalizasViewSet(viewsets.ViewSet):

    def create(self, request):
        baliza = Baliza.crear_baliza(request.data)
        queryset = Baliza.objects.all().values('mac')
        publicador = mqtt.Client()
        with open('config.json') as file:
            data = json.load(file)
            publicador.connect(data['ip'])
            publicador.publish("BeaconTracker/Beacons",
    payload=str(list(queryset)))
            publicador.disconnect()
            return JsonResponse({'baliza': str(baliza)})

    def list(self, request):
        queryset = Baliza.objects.all()
        serializer = BalizaSerializer(queryset, many=True)
        return Response(serializer.data)

    def retrieve(self, request, pk=None):
        queryset = Baliza.objects.filter(id=pk)
        serializer = BalizaSerializer(queryset, many=True)
        return Response(serializer.data)
```

9.3 Servizo de escaneamento | BLE.service

A funcionalidade principial dos rastrexadores consiste en escanear o espazo para capturar as mensaxes BLE emitidas polas balizas de xeito periódico. Para cumprir con esta función, implementouse un script en Python usando a librería *Bluepy* executándoo como servizo en segundo plano. Ademais, estes escaneamentos emítense mediante MQTT ao BackEnd. A continuación podemos ver a implementación.

```
import paho.mqtt.client as mqtt
from getmac import get_mac_address
from bluepy.btle import Scanner, DefaultDelegate
import time,json
mac_ble = get_mac_address(interface="wlan0")
cliente_mqtt = mqtt.Client()

def on_connect(client, userdata, rc,aux):
    client.subscribe("BeaconTracker/Beacons")

def on_message(client, userdata, msg):
    with open('config.json') as file:
        data = json.load(file)
        beacons=[]
        mensaje=msg.payload.decode("utf-8").strip(']['').split(', ')
        for beacon in mensaje:
            beacon=json.loads(beacon.replace("'",'"'))
            beacons.append(beacon['mac'])
        data['beacons']=beacons
    with open('config.json','w') as file:
        json.dump(data,file)

class ScanDelegate(DefaultDelegate):
    def __init__(self):
        DefaultDelegate.__init__(self)
        self.i=0

    def handleDiscovery(self,dev,isNewDev,isNewData):
        with open('config.json') as file:
            data = json.load(file)
            beacons=data['beacons']
            if beacons.count(dev.addr.upper())>0:
                try:
                    payload= json.dumps({"baliza":dev.addr.upper(),"rastreador":mac_ble.upper(),
                    "\acrshort{rssi}":dev.rssi,"fecha":time.time()})
                    cliente_mqtt.publish("BeaconTracker/"+mac_ble.upper()+"/Scan",payload=payload)
                except Exception as ex:
                    print("Scan System error:"+ str(ex))

if __name__=="__main__":
        cliente_mqtt.on_connect = on_connect
        cliente_mqtt.on_message = on_message
        with open('config.json') as file:
            configuration = json.load(file)
            cliente_mqtt.connect(configuration['ip'])
            cliente_mqtt.loop_start();
            time.sleep(0.25)
            cliente_mqtt.publish("BeaconTracker/"+
            mac_ble+"/Beacons",payload="Update beacons")
            scanner = Scanner().withDelegate(ScanDelegate())
            while(1):
                devices = scanner.scan(configuration['frec'])
```

9.4 Servizo de cálculo das posicións | LOC.service

A outra funcionalidade principial dos rastrexadores consiste en transformar os escaneamentos feitos nun intervalo de tempo, en localizacións. Isto débese calcular tamén de feito constante. Para cumprir con esta función, implementouse outro script en Python usando a librería *Localization* en executouse como servizo en segundo plano. A continuación podemos ver a implementación.

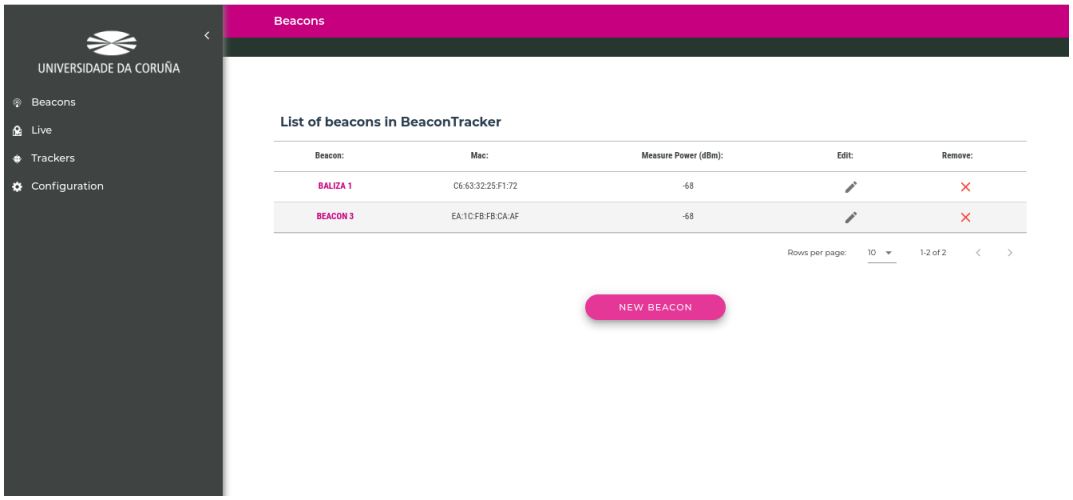
```
import time,requests,json
import localization as lx
header={'Content-Type': 'application/json'}
if __name__=="__main__":
    while(1):
        with open('config.json') as file:
            configuration = json.load(file)
            postions=[]
            for beacon in configuration['beacons']:
                payload = json.dumps({
                "start":time.time()-configuration['tiempo'],
                "end":time.time(),'mac':beacon})
                res = requests.get('http://'+configuration['ip']+
                ':9595'+'/api/scans/',headers=header,data=payload)
                response = res.json()

    punto=lx.Project(mode=response['mode'],solver=response['solver'])
                t,label = punto.add_target()
                if len(response['escaneos'])>=1:
                    if len(response['escaneos'])==1:
                        payload=json.dumps({"mac":beacon,
                        "latitud":response['escaneos'][0]
                        ['localizacion']['localizacion__latitud'],
                        "longitud":response['escaneos'][0] /
                        ['localizacion']['localizacion__longitud'],/
                        "radio":response['escaneos'][0]['distance']})
                        requests.request('POST',url='http://'+
                        configuration['ip']+':9595'+'/api/position/',
                        headers=header,data=payload)
                    else:
                        for scan in response['escaneos']:
                            punto.add_anchor(scan['rastreador'],
                            (scan['localizacion']['localizacion__latitud'],
                            scan['localizacion']['localizacion__longitud']))
                            t.add_measure(scan['rastreador'],
                            scan['distance'])
                        punto.solve()
                        payload=json.dumps({"mac":beacon,"latitud":t.loc.x,
                        "longitud":t.loc.y})
                        requests.request('POST',url='http://'+
                        configuration['ip']+':9595'+ '/api/position/',
                        headers=header,data=payload)
        time.sleep(configuration['tiempo'])
```

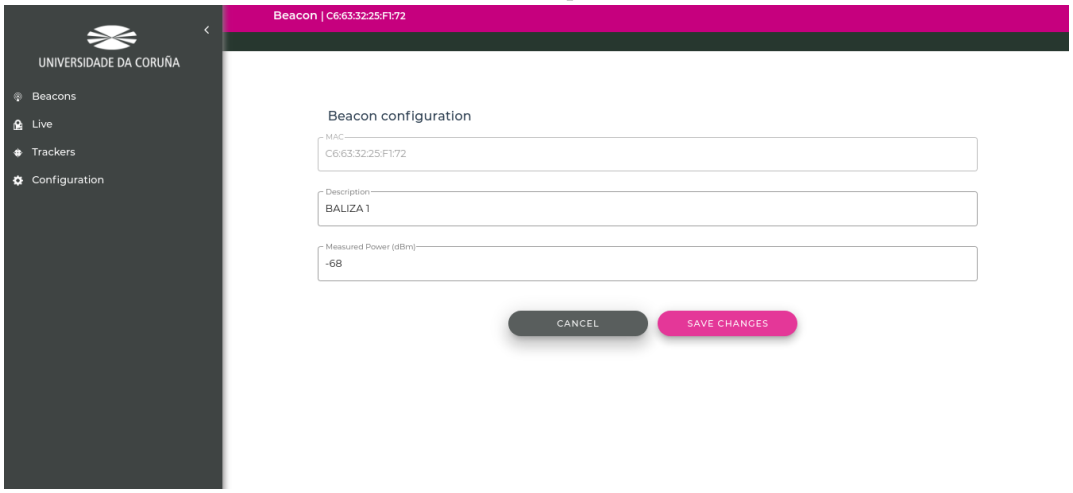
9.5 Implementación da interface

Para a interface fíxose un deseño na sección 8.5. Nela expoñiamos que necesitamos unha vista para a xestión das balizas, unha para a xestión dos rastrexadores, outra para a configuración e por suposto 2 vistas para a localización en tempo real e nun intervalo de tempo. A

continuación podemos ver como quedaron as vistas implementadas con Vue.js.

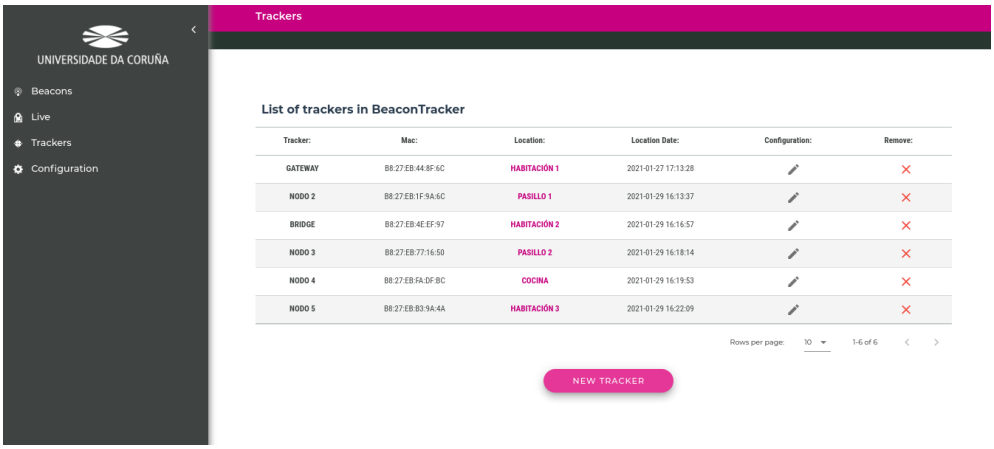


(a) Vista da lista das balizas que se están a escanear.

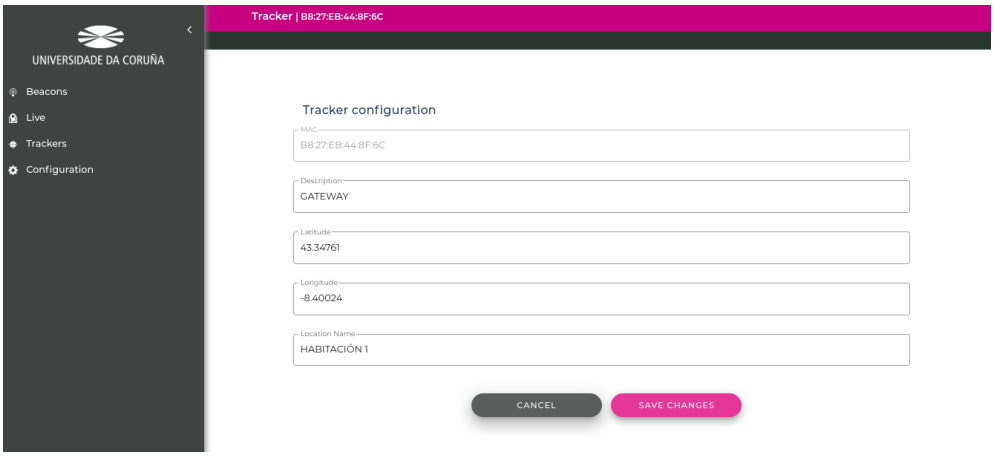


(b) Vista de creación/edición das balizas.

Figura 9.3: Vistas da xestión das balizas na interface creada.



(a) Vista da lista dos rastrexadores incluídos no sistema na interface creada.



(b) Vista de creación/edición dos rastrexadores na interface creada.

Figura 9.4: Vistas da xestión dos rastrexadores na interface creada.

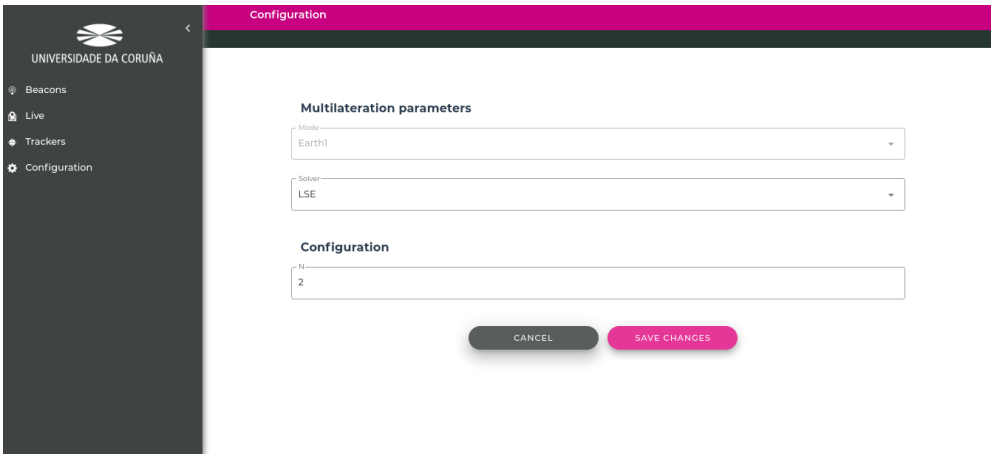
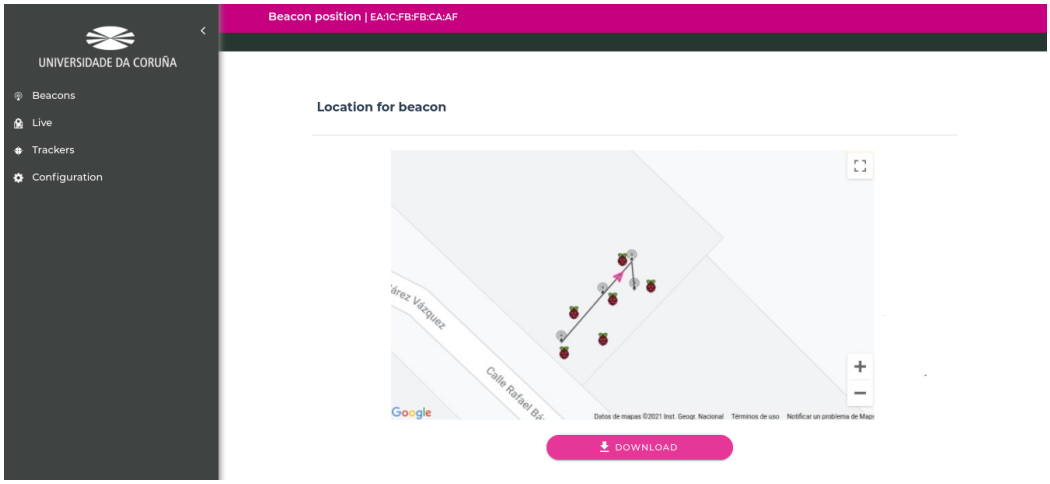
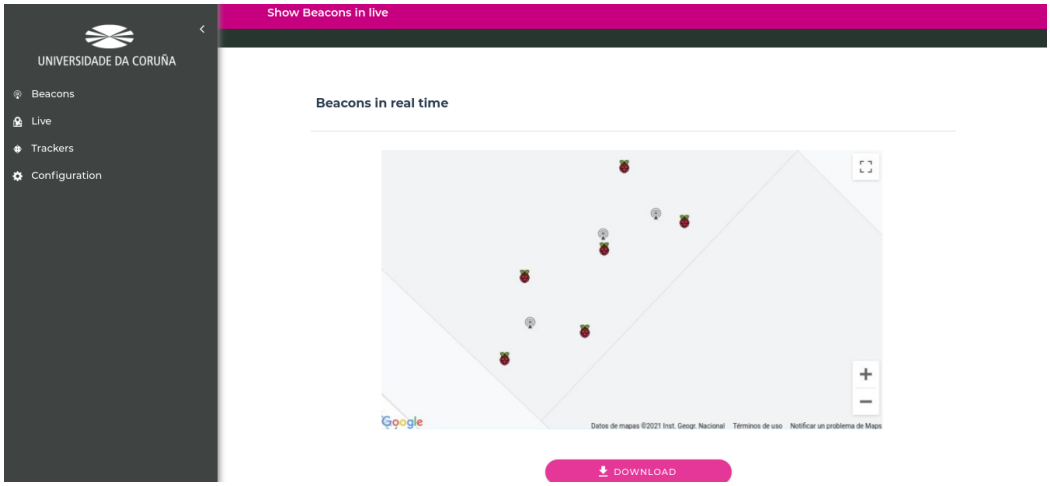


Figura 9.5: Vista para o configuración na interface creada.



(a) Vista dunha baliza nun intervalo de tempo na interface creada.



(b) Vista das balizas en tempo real.

Figura 9.6: Vistas da localización das balizas na interface creada.

Capítulo 10

Probas

Neste capítulo móstranse e explícanse os resultados das probas realizadas para a validación das funcionalidades desenvoltas no sistema. En primeiro lugar, realizáronse as probas de conexión da rede e da independencia de todo o sistema. En segundo lugar, procedeuse a calibrar os dispositivos empregados. E, por último, móstranse as probas de localización das balizas no entorno, tanto estática como dinámica.

10.1 Rede de malla

Os dispositivos do noso sistema están conectados mediante unha rede de malla. O obxectivo desta primeira proba é demostrar que aínda que o número de dispositivos aumente de forma considerable, de forma que moitos dos nodos non se poidan conectar de forma directa, a xestión da rede permite que calquera par de nodos se poidan conectar sen problemas a través da malla. En concreto, que todos os nodos se podan conectar ao nodo que xestiona todo o sistema.

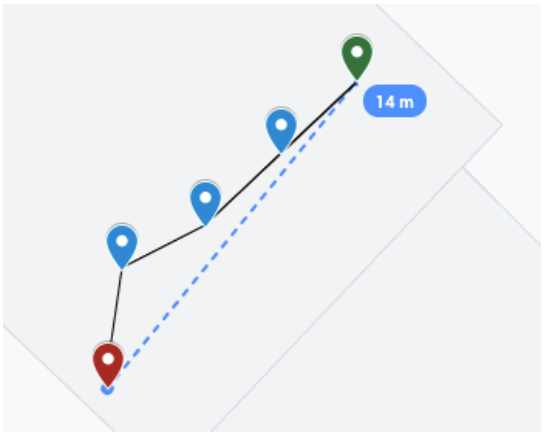


Figura 10.1: Situación dos nodos na proba da rede malla.

```
pi@nodo3:~ $ ifconfig
bat0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1468
        inet 192.168.199.15  netmask 255.255.255.0  broadcast 192.168.199.255
        inet6 fe80::5957:2bcb:9444:7dce  prefixlen 64  scopeid 0x20<link>
        ether 66:ba:e6:eb:43:35  txqueuelen 1000  (Ethernet)
        RX packets 271  bytes 19288 (18.8 KiB)
        RX errors 0  dropped 2  overruns 0  frame 0
        TX packets 177  bytes 20052 (19.5 KiB)
        TX errors 0  dropped 4 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 116  bytes 7677 (7.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 116  bytes 7677 (7.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 169.254.23.59  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::ba27:ebff:fe77:1650  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:77:16:50  txqueuelen 1000  (Ethernet)
        RX packets 12035  bytes 802274 (783.4 KiB)
        RX errors 0  dropped 36  overruns 0  frame 0
        TX packets 4349  bytes 457820 (447.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(a) IP e MAC do *nodo3*, usado para a proba da conexión da malla.

```
pi@gw1:~ $ sudo batctl n
[B.A.T.M.A.N. adv 2019.4, MainIF/MAC: wlan0/b8:27:eb:44:8f:6c (bat0/fe:0c:43:14:
e9:66 BATMAN_IV)]
IF             Neighbor              last-seen
        wlan0     b8:27:eb:1f:9a:6c    10.970s
        wlan0     b8:27:eb:4f:53:ce     3.200s
        wlan0     b8:27:eb:4e:ef:97     0.300s
```

(b) Comprobación de que *gw1* e *nodo3* non son veciños.

```
pi@gw1:~ $ ping 192.168.199.15
PING 192.168.199.15 (192.168.199.15) 56(84) bytes of data.
64 bytes from 192.168.199.15: icmp_seq=1 ttl=64 time=96.7 ms
64 bytes from 192.168.199.15: icmp_seq=2 ttl=64 time=487 ms
64 bytes from 192.168.199.15: icmp_seq=3 ttl=64 time=63.9 ms
64 bytes from 192.168.199.15: icmp_seq=4 ttl=64 time=164 ms
64 bytes from 192.168.199.15: icmp_seq=5 ttl=64 time=37.1 ms
64 bytes from 192.168.199.15: icmp_seq=6 ttl=64 time=63.7 ms
64 bytes from 192.168.199.15: icmp_seq=7 ttl=64 time=25.3 ms
^C
--- 192.168.199.15 ping statistics ---
8 packets transmitted, 7 received, 12.5% packet loss, time 17ms
rtt min/avg/max/mdev = 25.276/133.990/487.358/150.339 ms
```

(c) Ping realizado dende *gw1* a *nodo3* sin ser veciños.

Figura 10.2: Proba de funcionamento da rede coa topoloxía de malla mediante Ping.

```
pi@nodo3:~ $ mosquitto_pub -h "192.168.199.1" -t test -m "Hello World"
```

(a) IP e MAC do *nodo3*, usado para a proba da conexión da malla.

```
pi@gw1:~ $ mosquitto_sub -h localhost -t test
Hello World
```

(b) Comprobación de que *gw1* e *nodo3* non son veciños.

Figura 10.3: Proba de funcionamento da rede coa topoloxía de malla mediante MQTT.

Para esta proba colocamos dous nodos a unha distancia entre eles o suficientemente grande como para que non sexan nodos veciños. A distancia entre os dous nodos foi de 14 metros (ver imaxe 10.1). Para comprobar que os dous nodos non son veciños utilizamos o comando **sudo batctl n** nalgún deles para comprobar que efectivamente non o son.

Unha vez comprobado que non son veciños, podemos realizar varias probas para comprobar que cando queremos facer unha comunicación entre eles, o conseguimos mediante a rede de malla:

- A primeira comprobación, foi realizar un ping dende unha das máquinas a outra en cuestión. Ao facelo (figura 10.2) vemos que responde afirmativamente, polo que podemos garantir a comunicación entre elas.

- A segunda proba que se fixo foi facer unha comunicación mediante MQTT con Mosquitto para ver que recibe as publicacións correctamente. Ao facelo (figura 10.3) comprobamos que efectivamente podemos transmitir datos coa certeza de que serán recibidos.

10.2 Independencia da rede

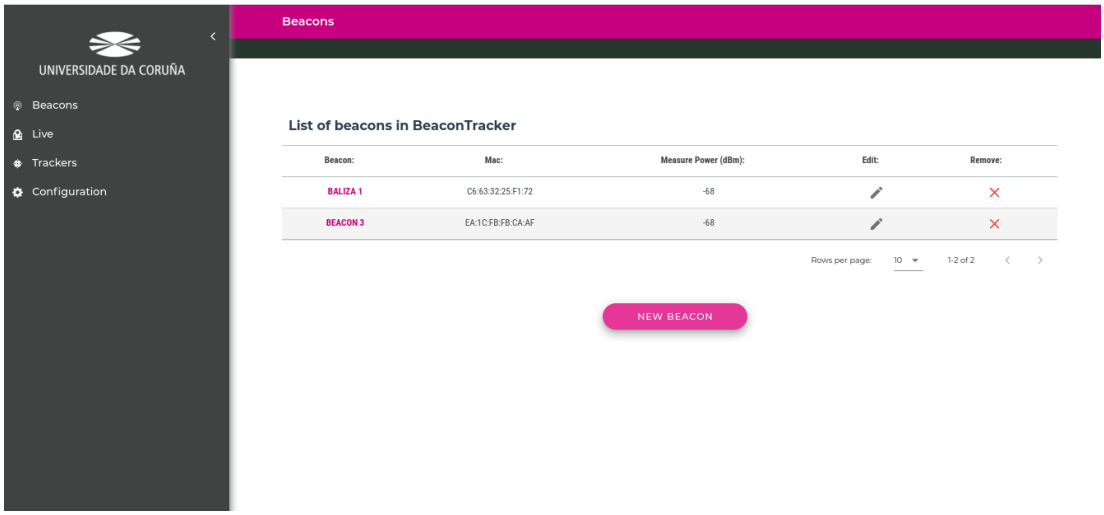
O obxectivo desta proba consiste en demostrar que a nosa rede malla pode operar sen a necesidade de estar conectada a unha rede xa existente ou a internet. Polo tanto, sería autónoma e podería despregarse en calquera edificio ou recinto.

Para esta demostración despregamos o noso sistema sen que a porta de enlace tivera conexión a outra rede externa. Unha vez despregado probamos a usar a nosa interface e comprobar que todas as funcionalidades operan correctamente.

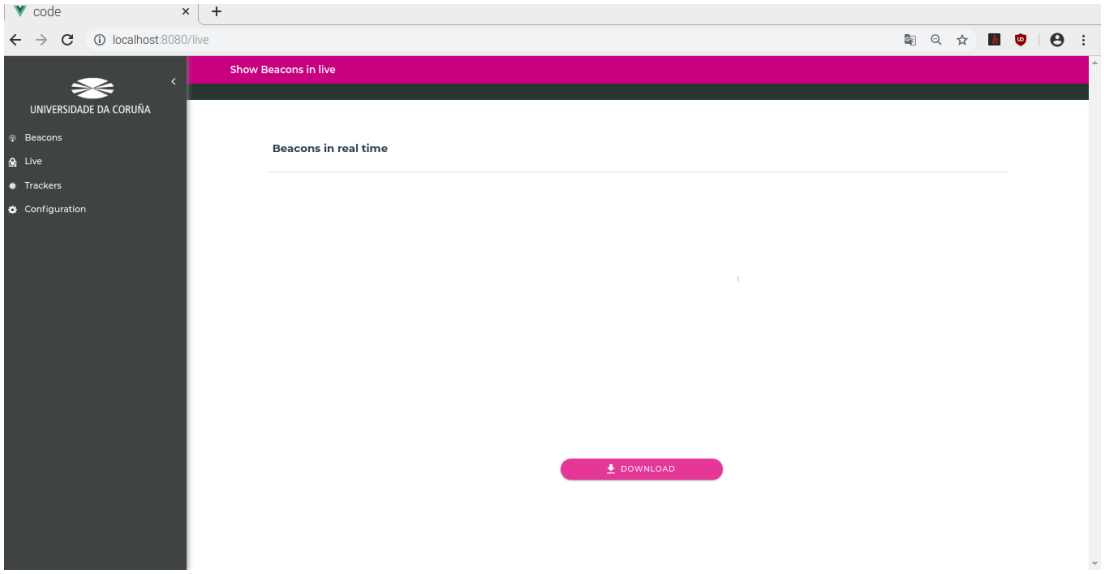
Na imaxe 10.4 podemos comprobar o funcionamento sen rede externa para o noso sistema. Como era previsto, salvo a utilidade de Google Maps, que ten unha dependencia externa, o funcionamento é normal.

O problema de non ter a capacidade de situar as balizas no mapa, debido a non ter acceso a Google Maps, foi unha decisión de desenvolvemento que se tomou pola sinxeleza que amosaba utilizar esta ferramenta e porque se primou a reducción no tempo de desenvolvemento.

Para solucionar este inconvinte, optouse por implementar unha utilidade que permitise descargar as posicións en formato CSV e así poder exportar as posicións e poder integralo no sistema ou ferramenta que se desexe. Nun futuro, se se quere evitar esta dependencia con Google Maps, podería plantexarse a importación estática de mapas, ou incluso valorar empregar planos ou imaxes das instalacións nas que se desexe despregar o sistema.



(a) Funcionamento da vista de balizas sen conexión a Internet.



(b) Funcionamento dos mapas sen conexión a Internet. As posicións devolveranse en formato CSV.

Figura 10.4: Vistas da interface sen conexión externa a rede.

10.3 Calibración

O obxectivo desta proba é axustar o máximo posible o cálculo da distancia das balizas mediante a función que relaciona o valor RSSI recibido polo rastrexador e esa distancia. Como se explica na sección 3.4 a fórmula usada para ese cálculo é:

$$Distancia = 10^{((PotenciaMedia - RSSI)/(10*N))} \tag{10.1}$$

onde podemos recordar que:

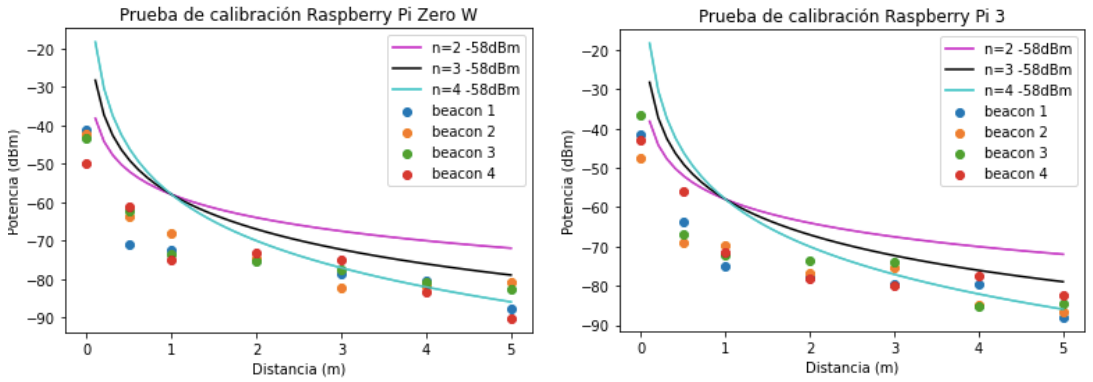
Figura 10.5: Situación das balizas na proba de calibración.

As distintas posicións das 4 balizas nas 7 configuracións (distancias) empregradas na calibración poden verse na imaxe 10.5.

- **Distancia** é a distancia medida en metros.
- A **Potencia Media** é unha constante calibrada de fábrica en cada dispositivo que indica cal é o RSSI esperado a unha distancia de 1 metro á baliza.
- **N** é a constante que depende do factor ambiental. Oscila nun rango entre 2 e 4, sendo 2 sistemas pouco conxestionado e 4 sistemas moi conxestionados.
- **RSSI** é o parámetro chave xa que é o indicador de intensidade do sinal recibido.

Para estimar esta relación leváronse a cabo varias probas de calibración. Mediante as cales obtivemos o valor N da constante do factor ambiental que nos permite calcular o RSSI en función da distancia. As probas realizáronse cos seguintes parámetros:

- Debido as restriccións pola COVID19 a calibración tivo lugar no corredor dun domicilio. O obxectivo era que todas as balizas tivesen "visión directa" cos rastrexadores.
- Só se calibrou un rastrexador de cada tipo: Raspberry Pi 3 e Raspberry Pi Zero W. Esta calibración depende moito do entorno e para despregar o sistema nunha infraestructura desexada quizáis habería que repetir a calibración.
- As 4 balizas se configuraron coa mesma frecuencia (1 Hz) e potencia media (-58 dBm).
- As distancias usadas para a calibración foron: 0, 0,5, 1, 2, 3, 4 e 5 metros (Fig. 10.6).
- Escaneouse o valor RSSI cada 2 segundos durante 5 minutos.



(a) Calibración da Raspberry Pi Zero W.

(b) Calibración da Raspberry Pi 3.

Figura 10.6: Resultados de calibracións de cada un dos dous tipos de Raspberry Pi.

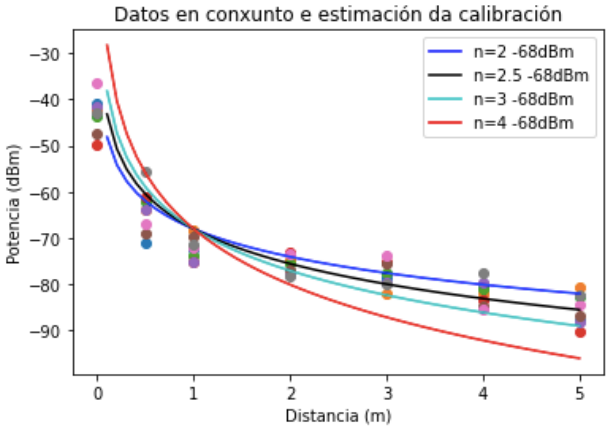


Figura 10.7: Resultados conxuntos de ambos tipos de Raspberry Pi nas probas de calibración.

A partir dos datos escaneados, calculouse o valor ambiental máis axustado. Para unha mellor análise fixeronse a probas dúas veces, unha coa Raspberry Pi 3 e outra coa Pi Zero Wi-fi como podemos observar na figura 10.6.

Como vemos, os datos extrapolados dos dous tipos de Raspberries (figura 10.7) son semellantes en canto á lectura do RSSI, o que nos permite deducir que non hai unha diferencia de usar unha ou outra. Tamén vemos que o RSSI lido está por debaixo do esperado en calquera das situacións ambientais que contemplamos, o que nos fai pensar que a potencia de trasmisión debe caer por algún motivo como pode ser que a batería ten pouca carga, fronte a isto asumimos unha perda de -10 dBm na trasmisión.

Superpoñendo os datos das Raspberry na imaxe 10.5 obtemos a calibración en conxunto e podemos asumir que os mellores valores para traballar na nosa ubicación son un **factor ambiental de 2,5**, (2,4 para a Raspberry Pi 3 e 2,6 Raspberry Zero W) e asumir que as balizas teñen unha potencia media de -68 dBm.

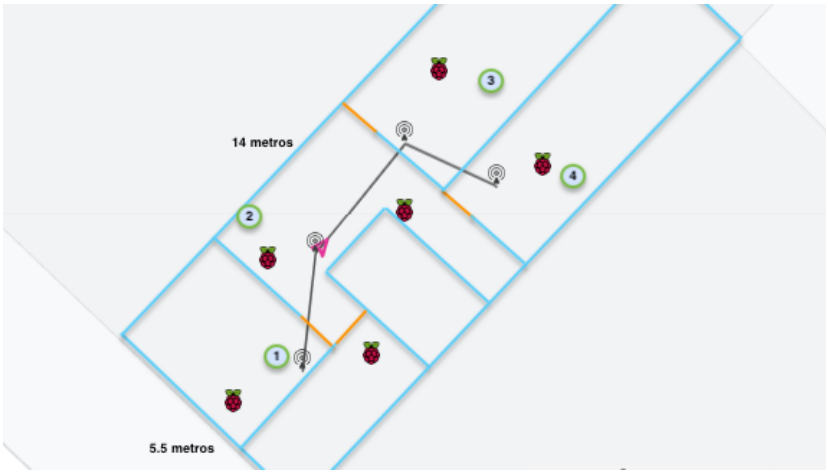


Figura 10.8: Proba de localización dunha baliza inmóvil polo noso sistema.

10.3.1 Comparativa co estado do arte

O artigo [10], fai referencia a un estudo semellante de cálculos de posicións con Raspberry. Neste artigo empregan tres modelos de calibración que relacionan o RSSI e a distancia. Hai que ter en conta que eles empregan outras balizas que envían sinais a 10Hz, mentres que nos o facemos a 1Hz. O aumentar a frecuencia das balizas poden empregar intervalos de tempo máis pequenos (eles empregan 1s), pero se recordamos o que dixemos na teoría (ver Sec.3.3), se aumentamos a frecuncia das balizas, aumentará o consumo. No noso caso optamos por un menor consumo e usando a mesma proporción, acadamos intervalos de 10 segundos.

Para o modelo de perda de traxectoria de distancia logarítmica, que é o que usamos nós, obtén un valor de N=2,6 para a constante que depende do factor ambiental en espazos interiores, o que nos permite validar os resultados acadados na calibración (N=2,5).

10.4 Localización estática

O obxectivo desta proba é analizar o erro que se produce no cálculo da posición con respecto a un punto de referencia. Para realizar esta análise comparamos as posicións calculadas polo sistema, a partir dos datos dos rastrexadores fronte a un punto de referencia simulado. Os parámetros da proba son os seguintes:

- A proba desenvolveuse nun domicilio debido ás limitacións da COVID19.
- No entorno dispuxéronse 6 rastrexadores e unha baliza BLE estática, que avaliaremos en 4 posicións diferentes (ver números na imaxe 10.8).
- As disposicións dos rastrexadores móstranse na imaxe 10.8 cos símbolos da Raspberry.

Táboa 10.1: Variación do cálculo da distancia segundo o RSSI coa calibración estimada.

RSSI (dBm)	Distancia (m) (Ec. 10.1)	Δ Distancia (m)
-48	0,158	-
-53	0,251	0,097
-58	0,398	0,107
-63	0,631	0,233
-68	1,000	0,369
-73	1,585	0,585
-78	2,512	0,927
-83	3,981	1,479
-88	6,310	2,329
-93	10,000	3,690
-98	15,849	5,849

Táboa 10.2: Datos do RSSI recollidos polos 6 nodos no punto de referencia 1.

	Intervalo 1			**Intervalo 2**			**Intervalo 3**			**Intervalo 4**			**Intervalo 5**		
NODO	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*
0	-65,2	6,3	4	-73,2	4,6	4	-75,3	0,4	3	-69,5	6,5	4	-69,5	8,5	2
1	-88,5	4,6	6	-93,5	0,7	4	-85,5	7,5	2	-88,0	7,3	3	-86,6	5,1	3
2	-69,2	5,1	4	-74,0	6,5	4	-73,1	5,4	6	-73,2	10,2	5	-76,8	3,6	5
3	-81,2	2,1	4	-84,6	2,2	3	-90,0	0,0	2	-78,0	3,0	4	-87,0	7,0	4
4	-	-	0	-88,7	5,7	4	-96,0	0,0	1	-86,2	4,7	4	-87,3	5,5	3
5	-63,0	5,5	4	-66,0	0,4	5	-62,4	6,1	5	-65,8	9,8	5	-72,0	1,3	3

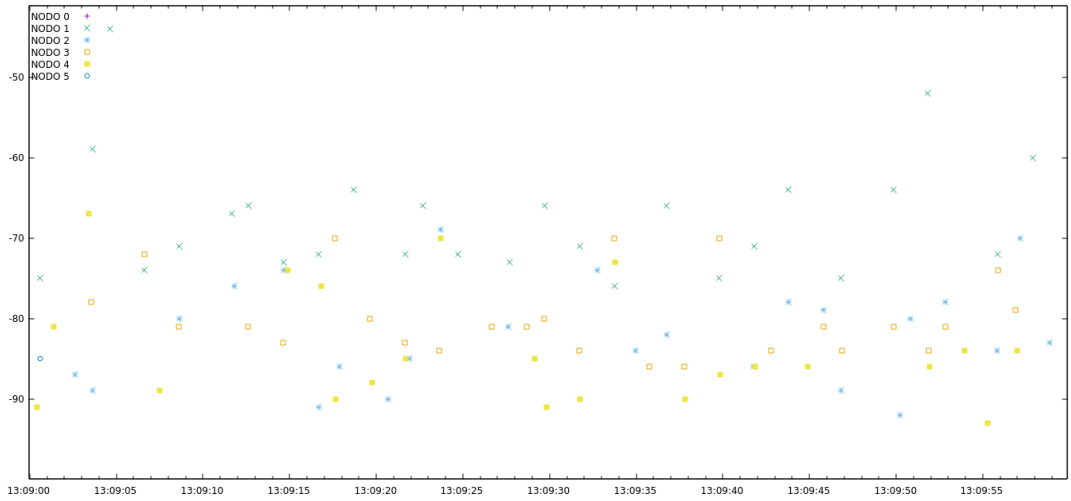
- Para cada posición de referencia da baliza (ver números na imaxe 10.8), gardáronse as medidas de todos os rastrexadores durante un intervalo de 1 minuto.

- Cada 10 segundos, e sempre que houbese suficientes datos, o sistema calculou a posición da baliza (amosada como unha traxectoria na imaxe 10.8 e como puntos na imaxe 10.9).

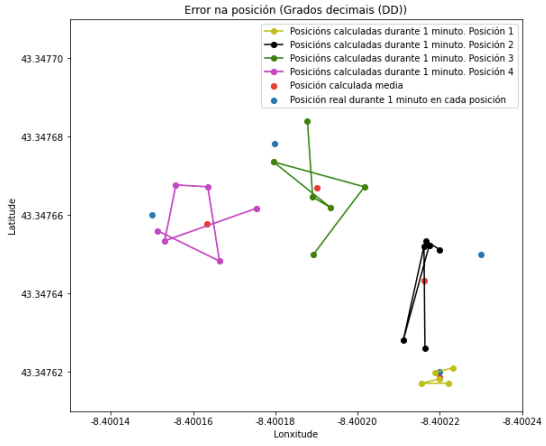
- Mesma configuración de rastrexadores e baliza que na calibración (Sec. 10.3).

Á vista dos resultados acadados (10.9), comprobamos que as posicións calculadas para cada posición de referencia fluctúa, o motivo desta variación débese a que o RSSI varía a pesar de que a baliza permanece estática.

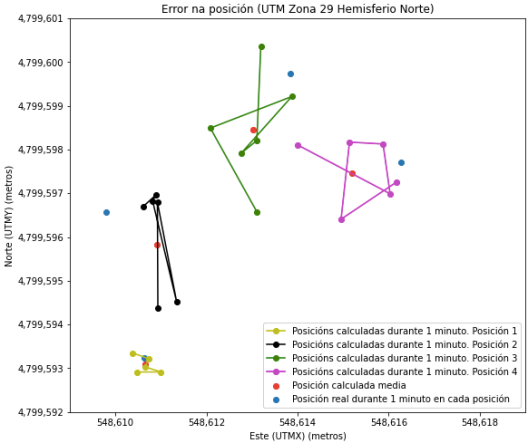
Para entender que influencia ten esta variación do RSSI no sistema (na imaxe 10.9a podemos ver todos os RSSI lidos na proba), debemos estudiar canto afecta ao cálculo da distancia da baliza para cada rastrexador e para iso analizaremos os puntos de referencia 1 e 3, que amosa o mellor e o peor resultado, respectivamente.



(a) RSSI medidos por todos os rastrexadores ou nodos do noso sistema.



(b) Posicións no plano decimal.



(c) Posicións no plano UTM expresado en metros.

Figura 10.9: Localización estática: comparativa entre as 4 posicións reais de referencia e as calculadas polo noso sistema cada 10 segundos nun intervalo de 1 minuto (6 valores).

10.4.1 Análise do RSSI no punto de referencia 1

Para valorar a influencia do RSSI elaborouse unha táboa que contén todos os datos recollidos polos rastrexadores nos intervalos de tempo do punto de referencia (Tab. 10.2). Como vemos, a desviación típica do RSSI fronte a media, sitúase nun rango de 0-10 dBm. Hai que ter en conta que cada nodo calcula a distancia a que está a baliza con respecto a el, polo que se calculamos canto afecta unha variación de 10 dBm coa formula que acadamos na calibración (Fig. 10.7). Na táboa 10.1 se mostra o erro que se produce en cada nodo.

O cálculo feito permítenos descubrir que o erro para unha diferencia de 10 dBm varía habitualmente entre 0,15 e 6,0 metros aproximadamente, xa que a partir de -95 dBm os rastrexadores xa non son capaces de detectar as balizas. Obsérvase tamén que para un RSSI ≥ -73 dBm, existe unha variación $\leq 0,6$ metros, mentres que para un RSSI < -73 dBm, existe unha variación ≥ 1 metro.

Agora que sabemos canto pode afectar unha variación do RSSI ao cálculo da distancia nun só nodo, podemos entender porque a posición calculada pode variar a pesar de que a baliza non se mova de sitio. Sería lóxico pensar que un menor desvío do RSSI implicaría un menor erro, pero hai que analizar a situación.

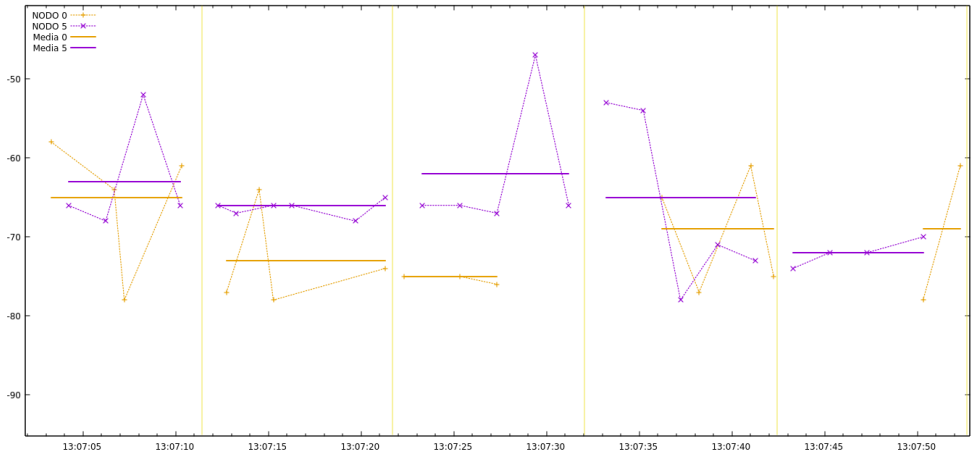
Nas figuras 10.10 detallamos os intervalos de dous dos rastrexadores que detectan a baliza para ver graficamente a desviación do RSSI. Como vemos, temos intervalos con pouca desviación (ex. intervalo 2) e outros cunha maior desviación (ex. intervalo 4). Entón para ver se existe unha correspondencia entre a desviación do RSSI e o erro na posición calculada, debemos comparar a desviación típica acumalada en cada un dos intervalos, e o erro que se produciu (Fig. 10.11).

Tras analizar os resultados podemos ver que, a pesar de que hai intervalos con menor desviación, isto non implica que o erro fronte o punto de referencia sexa menor. Polo menos neste caso no que se acadaron os mellores resultados cun erro máximo de 0,7 metros (Fig. 10.11). Para asegurarnos, faremos a análise na situación contraria, é dicir, no punto de referencia cos peores resultados (punto de referencia 3).

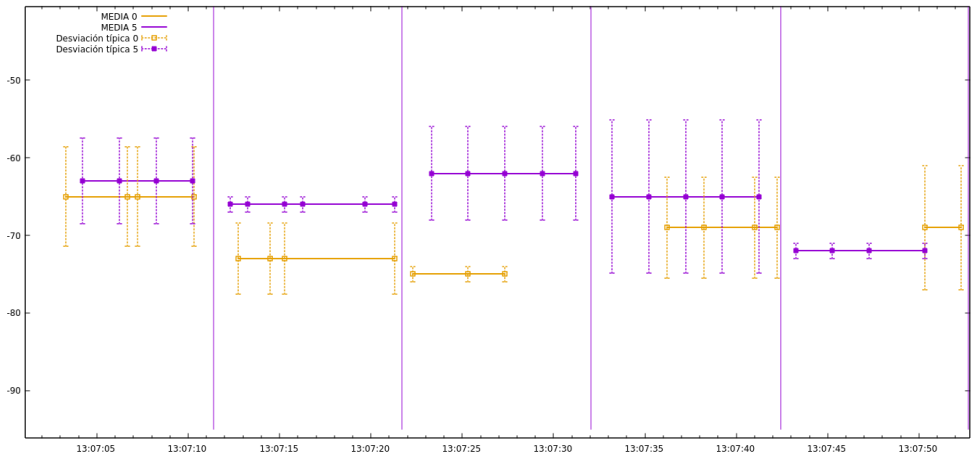
10.4.2 Análise do RSSI no punto de referencia 3

No punto de referencia 3 o erro producido no cálculo da posición foi o maior de todos (Figura 10.9). Analizar este caso pode axudarnos a entender mellor a desviación do RSSI. Tal como fixemos no anterior caso, na táboa 10.3 mostramos todos os datos recollidos para este punto de referencia. Na figura 10.12 vemos en detalle a variación do RSSI en dous nodos.

Observando as figuras vemos que hai intervalos cunha grande desviación do RSSI e outros máis estables. Loxicamente unha variación do RSSI implica un cambio na distancia calculada, tal e como vimos anteriormente (Táboa 10.1), pero os períodos de maior estabilidade non



(a) RSSI medidos polo nodo 0 e o nodo 5 e os seus valores medios en cada intervalo de 10 segundos.



(b) RSSI medidos polo nodo e os seus valores medios en cada intervalo de 10 segundos.

Figura 10.10: RSSI medidos na proba estática na posición de referencia 1 durante 1 minuto.

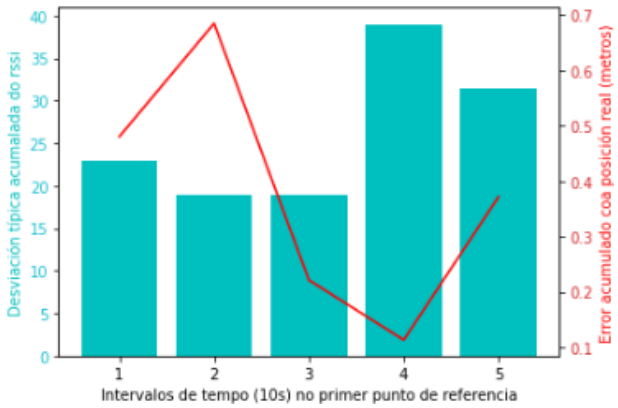
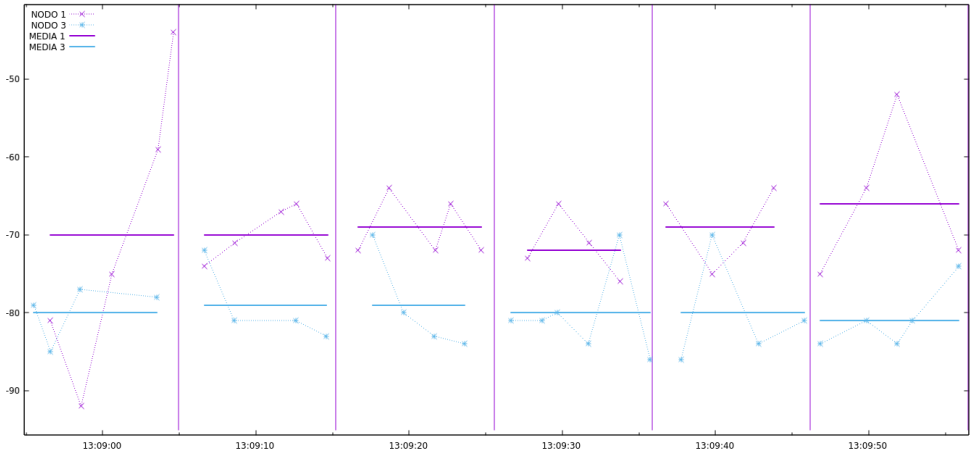


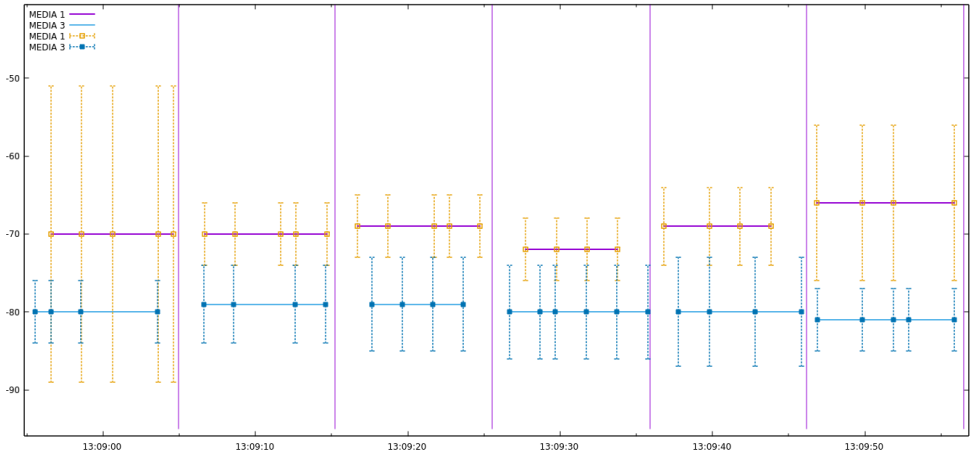
Figura 10.11: Comparativa entre a desviación típica do RSSI e o error da posición 1.

Táboa 10.3: Datos do RSSI recolidos por todos os nodos no punto de referencia 3.

Nodo	Intervalo 1			Intervalo 2			Intervalo 3			Intervalo 4			Intervalo 5			Intervalo 6		
	Med.	*Desv.*	#	*Med.*	*Desv.*	#	*Med.*	*Desv.*	#	*Med.*	*Desv.*	#	*Med.*	*Desv.*	#	*Med.*	*Desv.*	#
0	-87,0	-	1	-	-	0	-	-	0	-	-	0	-	-	0	-	-	0
1	-70,2	14,9	5	-70,2	2,9	5	-69,2	3,3	5	-71,5	3,0	4	-69,0	4,0	4	-65,7	7,7	4
2	-80,0	6,4	5	-76,6	2,2	3	-84,2	6,1	5	-79,6	3,7	3	-81,2	2,7	4	-84,6	4,7	5
3	-79,7	2,6	4	-79,2	3,6	4	-79,2	4,6	4	-80,3	3,5	6	-80,2	5,1	4	-80,8	2,7	5
4	-79,7	8,4	3	-81,5	7,5	2	-81,8	7,0	5	-84,7	5,8	4	-87,6	1,5	3	-87,6	3,5	3
5	-89,7	3,5	3	-	-	0	-	-	0	-	-	0	-	-	0	-	-	0



(a) RSSI medidos polo nodo 3 e os seus valores medios en cada intervalo de 10 segundos.



(b) RSSI medidos polo nodo 3 e os seus valores medios en cada intervalo de 10 segundos.

Figura 10.12: RSSI medidos na proba estática na posición de referencia 3 durante 1 minuto.

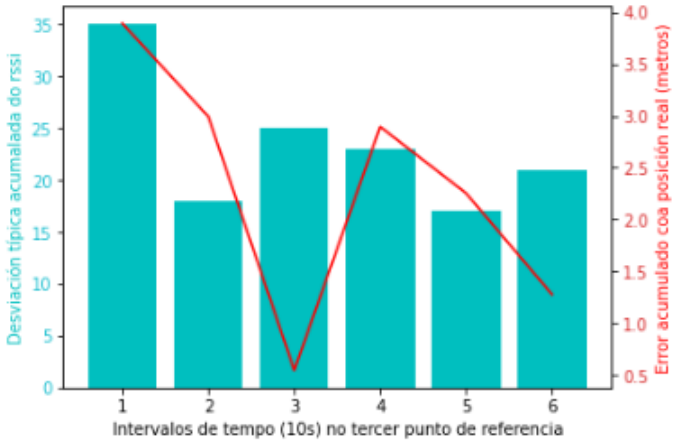


Figura 10.13: Comparativa entre a desviación típica do RSSI e o error da posición.

teñen porque corresponderse con medidas máis exactas ou precisas. Para ver se existe esta correlación, compararamos a desviación típica acumalada co erro que se produciu, tal e como fixemos no anterior punto de referencia. A figura 10.13 amosa esta comparativa onde se pode ver que o erro acadado chega aos 4 metros, o que confirma que é o peor caso dos considerados.

Tras analizar os resultados, volvemos a ver que a pesar de que hai intervalos con menor desviación, isto non implica maior erro fronte ao punto de referencia. Entón, que conclusión podemos sacar da análise? Como non existe unha correspondencia entre a desviación do RSSI e a veracidade de que ese intervalo sexa o correcto, non podemos asumir que os intervalos con menor desviación sexan mellores para o cálculo da posición. Polo tanto, asumir que a media das posicións calculadas, no espazo de tempo que consideremos, é a mellor opción para aumentar a fiabilidade do sistema, parece o máis razoable.

10.4.3 Comparativa dos resultados co estado do arte

No artigo [10], asumen tres posibles escenarios en interiores empregando 4 Raspberry como rastrexadores. No escenario máis pequeno (5x5 metros cadrados), realizan 3 probas nun escenario de 2x3 metros cadrados "efectivos", e amosan un erro de 3-4 m, aínda que co seu método logran reducilo a 2. No escenario máis grande (18x18 metros cadrados), realizan 6 probas nun escenario de 7x12 metros cadrados "efectivos", e amosan un erro de 7-9.5 m, co seu método logran reducilo a 5.

O noso caso é un escenario de 14x5.5 metros cadrados, dos cales 9x5 son os metros cadrados "efectivos". Empregamos 6 rastrexadores, pero temos o inconvinte de que ter paredes que inflúen moi negativamente no sinal. O erro no noso escenario é de 3-4 metros no peor caso, e de 1 m no mellor. Polo tanto, os nosos resultados son comparables co estado da técnica.

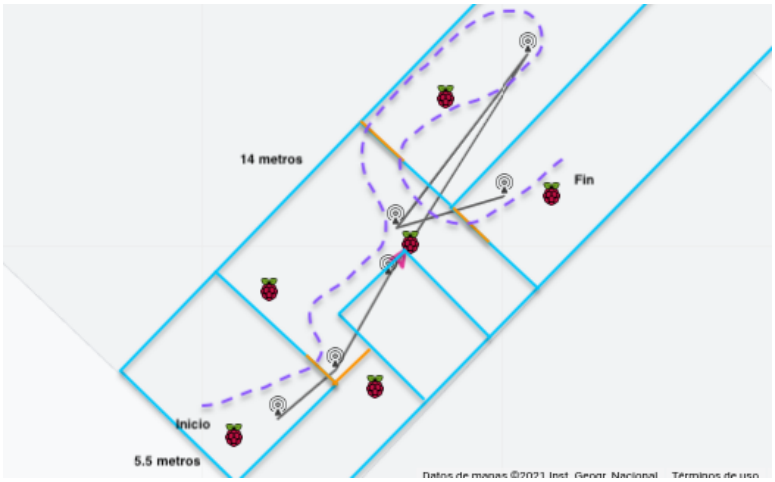


Figura 10.14: Proba de localización dunha baliza en movemento no noso sistema.

10.5 Localización dinámica

Para probar que ocorrería se a baliza estivese en movemento, é interesante analizar como se comportaría o sistema neste caso. Para isto comparouse a traxectoria calculada polo sistema coa traxectoria da baliza simulada. Os parámetros para realizar a proba foron os seguintes:

- A configuración do entorno para a proba (rastrexadores, balizas, etc,) foi igual que na proba estática (Sec. 10.4).
- A proba realizouse a unha velocidade constante (0,5 m/s aprox.) durante 1 minuto, seguindo a traxectoria amosada na figura 10.14 e percorrendo uns 30 metros.

Para a análise dos resultados, seguimos a mesma metodoloxía que na proba estática: preparamos a táboa 10.4 cos resultados de cada rastrexador e a figura 10.15 coa evolución dos RSSI en cada intervalo.

Observando estes intervalos, vemos que o RSSI varía segundo nos movemos. Esta variación debería ser constante, pero como vimos anteriormente (ver Sec.10.4), o RSSI fluctúa, o que produce desviacións.

Na imaxe 10.14, móstrase o resultado da proba no sistema. Como vemos, o sistema é capaz de seguir a traxectoria da baliza.

Para analizar o resultado, simulamos a traxectoria nun plano UTM (Fig.10.16 e Fig.10.17) e comparámola coa traxectoria calculada. Esta análise permítenos descubrir, que aínda que o sistema é capaz de seguir a traxectoria, prodúcese unha desfase de tempo respecto á traxectoria simulada.

Na gráfica 10.18, podemos ver o erro que se produce no cálculo da posición fronte a posición simulada. Hai que ter en conta que o desfase inflúe negativamente cálculo da posición.

Táboa 10.4: Datos do RSSI recollidos nos 6 nodos na proba dinámica.

	Intervalo 1			**Intervalo 2**			**Intervalo 3**			**Intervalo 4**			**Intervalo 5**			**Intervalo 6**		
Nodo	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*	*Med.*	*Desv.*	*#*
0	-59,8	7,1	6	-80,2	2,8	4	-86,0	0,0	1	-	-	0	-95,0	0,0	1	-	-	0
1	-87,3	5,6	3	-84,8	3,9	5	-75,4	5,6	5	-61,6	9,12	5	-45,3	10,4	3	-73,5	9,5	4
2	-72,2	5,3	5	-79,2	4,1	5	-71,0	6,0	4	-81,8	4,16	5	-64,4	4,7	5	-88,0	2,0	2
3	-81,0	4,0	3	-84,0	1,0	2	-73,8	3,7	5	-75,6	11,0	5	-73,4	4,3	5	-74,5	5,0	4
4	-	-	0	-90,0	3,0	2	-77,0	6,0	3	-73,0	5,6	5	-73,8	9,3	4	-65,8	5,9	5
5	-78,0	4,0	6	-71,0	3,0	4	-79,5	2,8	6	-	-	0	-86,7	5,8	3	-	-	0

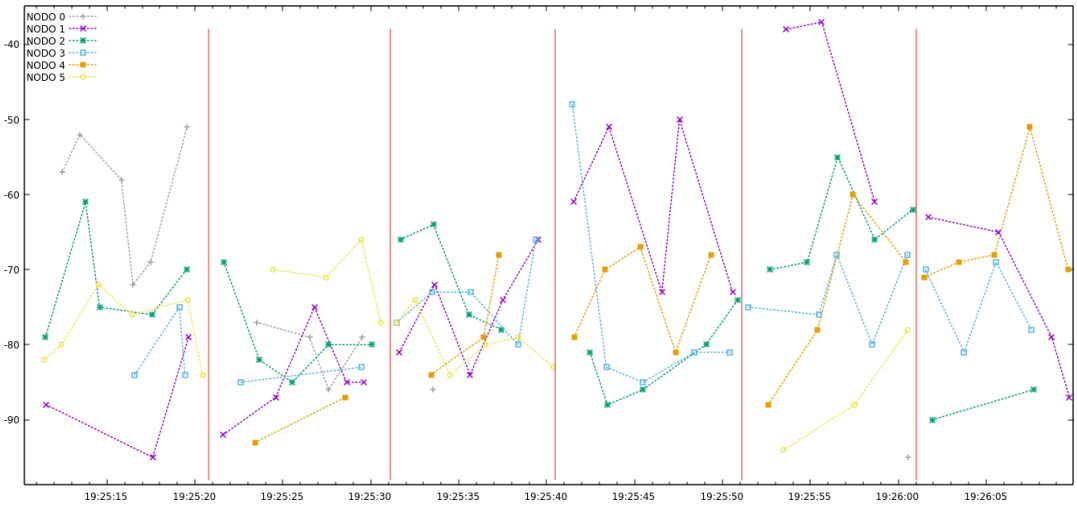
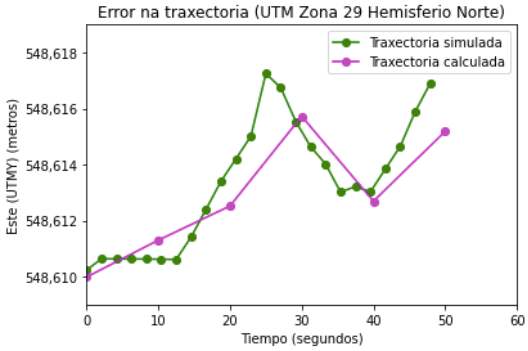


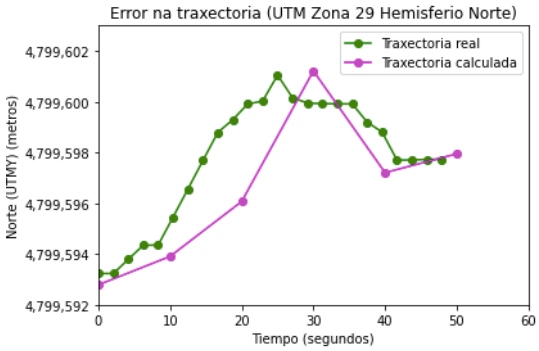
Figura 10.15: Variación do RSSI na proba de localización dinámica.

Para reducir o desfase poderíase aumentar a frecuencia de mostraxe, o que implicaría aumentar a frecuencia das baliza, e consecuentemente un maior consumo (ver Sec.3.3).

Á vista dos resultados, o noso sistema é capaz de calcular traxectorias con estes parámetros, a razón de intervalos de 10 segundos e cun erro $< 6,5$ metros.



(a) Variación da distancia o Este respecto o tempo.



(b) Variación da distancia o Norte respecto o tempo.

Figura 10.16: Comparativa desglosada da proba de localización dinámica.

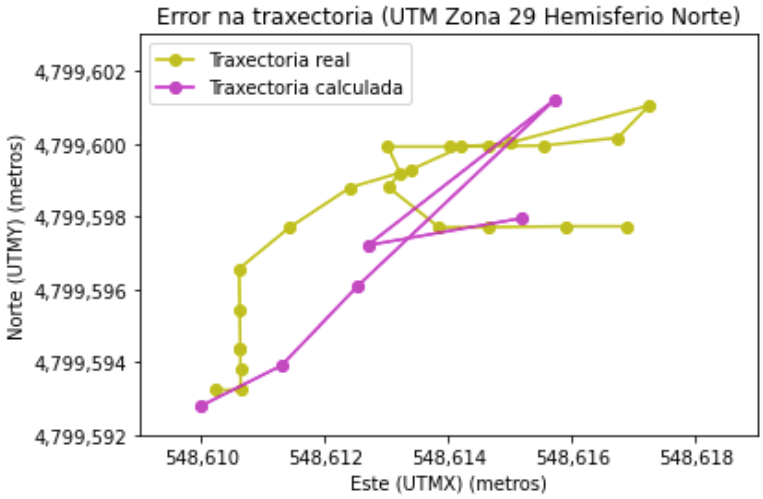


Figura 10.17: Comparativa conxunta da proba de localización dinámica.

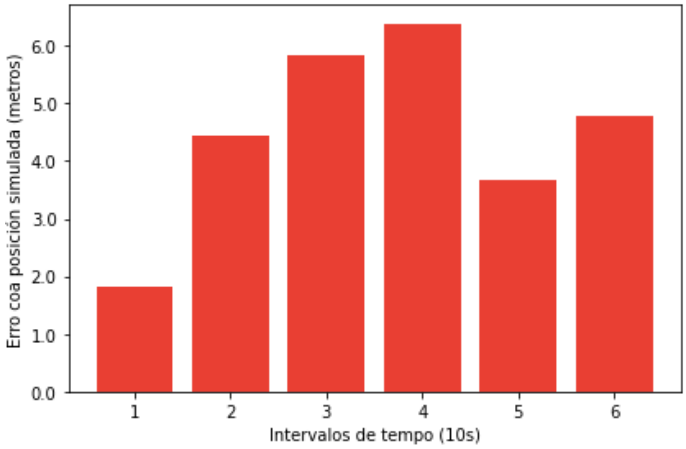


Figura 10.18: Erro producido na proba de localización dinámica.

Capítulo 11

Conclusións

Este é o derradeiro capítulo da memoria, onde se presentará a situación final do traballo, as leccións aprendidas, a relación coas competencias da titulación en xeral e a mención en particular e as posibles liñas futuras.

11.1 Conclusións

O obxetivo desde proxecto era crear un sistema de localización en espazos interiores intentando ofrecer melloras a traballos anteriores. Á vista dos resultados obtidos podemos decir que conseguimos o seguinte:

A nivel funcionalidades

- Creouse un sistema capaz de localizar balizas BLE en interiores, tanto de forma estática como de forma dinámica, empregando hardware de baixo custo.
- O sistema pódese despregamento facilmente en calquer entorno, xa que no require infrastrutura adicional.
- Implementouse unha rede en malla que pode configurarse co número que se quera de nodos, o que nos permite expandir a rede e, en consecuencia, mellorar a precisión en sitios grandes ou complexos.
- Pode haber múltiples nodos Bridge para conectarse o sistema.
- Empregouse o algoritmo de multilateración, polo que que todos os nodos que detecten as balizas inflúen no cálculo e, polo tanto, conseguiremos unha maior precisión do noso sistema.
- Creouse unha API REST que permite modularizar a lóxica de negocio e en consecuencia podríase integrarse noutras aplicacións.

- Desenvolveouse unha interface gráfica de usuario amigable, sinxela e que permite empregar a funcionalidades expostas pola API REST mencionada.
- A modularidade do sistema permite traballar por separado no sistema e así poder modificar a parte que máis nos interese sen ter que modificar as outras.

A nivel académico

Con relación á titulación este traballo permitiu o seguinte:

- Mellorar o traballo en equipo ao traballar conxuntamente co titor para desenvolver o proxecto.
- Mellorar a capacidade de detección de problemas, por exemplo os relacionados coa detección das balizas, e coa variación das lecturas do RSSI e o despregamento da rede en malla.
- Mellorar a capacidade de análise e síntese o traballar con novas tecnoloxías coas que non se estaba familiarizado.
- Mellorar a capacidade para organizar e planificar seguindo o método SCRUM.
- Mellorar a capacidade para xerar novas ideas para poder mellorar o rendemento da aplicación e aportar funcionalidades.
- Mellorar a análise da realidade, diagnosticar problemas, ou formular e implantar solucións baseadas no coñecemento e orientadas ao ben común.
- Orientarse a un mundo laboral onde ten moita importancia a investigación, a innovación e o desenvolvemento tecnolóxico no avance socioeconómico e cultural da sociedade.

A nivel da mención

- Mellorar a capacidade de deseño e construción de sistemas basados en comunicacións.
- Mellorar a capacidade de análise para evaluar e programar arquitecturas.
- Mellorar o deseño e implementación de software de sistema e de comunicacións como por exemplo o *broker* MQTT.
- Mellorar a capacidade para programar, analizar e evaluar sistemas de tempo real.
- Mellorar a capacidade para evaluar e configurar infraestructuras informáticas para a execución de servizos.

11.2 Liñas futuras

Como retos principais proponse:

- Mellorar a escalabilidade, ben mediante unha base de datos distribuída que sexa compatible coa rede en malla, ben con técnicas de tolerancia a fallos.
- Estudar un sistema de localización baseado na tecnoloxía ZigBee.
- Distribuír os cálculos entre varios dispositivos en caso de aumentar moito a extensión do sistema.
- Aplicar un filtro de partículas para mellorar o cálculo das posicións empregando lecturas anteriores.
- Mellorar a precisión mediante novos algoritmos ou hardware máis mellor (antenas, etc.).
- Integrar o sistema cunha aplicación Android, como interface gráfica de usuario.
- Automatizar o despregamento do sistema ao conectar os nosos rastrexadores (detección e configuración automática).
- Crear unha monitorización automática do noso sistema, con autoconfiguración e alertas en casos como caídas ou desconexións dalgún rastrexador.

Apéndices

Apéndice A

Instalación e uso

Este capítulo, inclúe material adicional para a instalación do sistema e un manual de uso.

A.1 Requisitos

Para os rastrexadores debe terse unha Raspberry co sistema operativo Raspbian que teña instalado o seguinte software:

- Node.js e npm (**sudo apt-get install nodejs npm**)
- libreria *libgeos*. (**sudo apt-get install libgeos-dev**)
- libreria *libatlas*. (**sudo apt-get libatlas-base-dev**)
- *virtualenv*. (**sudo pip3 install virtualenv**)
- *broker* MQTT. (**sudo apt-get install mosquitto mosquitto-clients**)

A.2 Instalación

Para a instalación do sistema debemos seguir os seguintes pasos:

A.2.1 Creación da rede *bat0*

Para todos os nodos debemos configurar a rede tal e como se explica na sección 9.1.1.

A.2.2 Instalar o BackEnd

1. Co código do proxecto en Django debemos crear un proxecto Django no *Gateway* e crear un entorno virtual (**virtualenv .venv**)

2. Unha vez activado este entorno (**source .venv/bin/activate**) debemos instalar os requisitos (**pip3 install -r requirements.txt**).

3. Realizar as migracións da base de datos (**Python3 manage.py makemigrations** e **Python3 manage.py migrate**).

4. Crear un servizo para o script de localización (*loc.py*)

5. Levantar o sistema (**Python3 manage.py runserver 0.0.0.0:9595**).

A.2.3 Instalar o servizo BLE.service

En todos os nodos debemos crear un servizo para o script *ble.py* que realiza os escaneos dos dispositivos. Para que este servizo funcione debe estar levantado o BackEnd debido a que se conecta mediante un *broker* MQTT.

A.2.4 Instalar o FrontEnd

Para instalar o FrontEnd solo necesitamos copiar o noso proxecto de FrontEnd no Gateway e facer un **npm install**. Para levantalo só hai que executar **npm run serve**.

A.3 Guía rapida

Para usar a interface só debemos acceder á url do host do servidor **192.168.199.1:8080**. Unha vez ahí temos:

- Vista Balizas: mostra as balizas a escanear. Se prememos no nome iremos a vista do histórico.
- Vista Rastrexadores: mostra os rastrexadores levantados no sistema e que están operativos para ler. Se queremos incluír algún, hai que engadilo no sisitema antes de encendelo, se non o escaneo non funcionará.
- Vista Configuración: permite cambiar os parámetros de configuración do cálculo da posición.
- Vista Tempo real: mostra en tempo real a localización das balizas no mapa.

Apéndice B

Repositorio do sistema

Este capítulo inclúe a dirección url do repositorio onde se encontran o software do sistema.

B.1 Contido do repositorio

- **BACKEND**. Inclúe o código fonte do proxecto en Django do BackEnd.
- **FRONTEND**. Inclúe o código fonte do proxecto en Vue.js da interface gráfica.
- **Beacontracker**. Inclúe o código fonte dos scripts Python empregados.
- **services**. Inclúe un exemplo do código do servizo para os scripts Python creados.

B.2 Enlace ao repositorio

O software do sistema está accesible no OneDrive institucional da UDC é pode ser consultado seguindo o enlace [41].

Relación de Acrónimos

ACID Atomicity, Consistency, Isolation and Durability. 16

API REST Application Programming Interface Representational State Transfer. 1, 3, 15, 16, 18, 33, 40

BLE Bluetooth Low Energy. 1, 2, 6–9, 11, 16, 17, 19, 21, 29, 31, 38, 53, 65, 75

GPS Global Positioning System. 1, 5, 6, 12

HTML HyperText Markup Language. 15, 50

HTTP Hypertext Transfer Protocol. 15, 19, 41

IoT Internet of things. 9

JSON Javascript Object Notation. 15, 41, 43

MQTT Message Queuing Telemetry Transport. 1, 19, 30, 33, 40, 42, 53, 76, 81, 82

MTV Model Template View. 50

MVC Model View Controller. 50, 53

RSSI Received Signal Strength Indicator. 1, 2, 8, 10–13, 16, 19, 21, 24, 39–41, 43, 62–66, 68, 71, 72

Glosario

BackEnd Cando falamos de BackEnd referímonos ao interior das aplicacións que viven no servidor e ao que a miúdo se lle denomina "o lado do servidor". O BackEnd dun sitio web consiste nun servidor, unha aplicación e unha base de datos e encárgase de tomar os datos, procesar a información e envíala ao usuario. 15, 30, 32, 33, 39–42, 47, 50, 53, 82, 83

Bluetooth Bluetooth é unha especificación industrial para redes sen fíos de área personal. 1, 6, 7, 9

Bridge Un Bridge é unha ponte a outra rede. Funciona na capa de ligazón de datos e conecta dous LAN diferentes que traballan no mesmo protocolo. Ademais, na ponte, o formato do paquete non se cambia. 34, 38, 47, 75

C C é unha linguaxe de programación de propósito xeral. 16

EndPoints En poucas palabras, un EndPoint é un extremo dunha canle de comunicación. Para as API, un EndPoint pode incluír unha URL dun servidor ou servizo e para cada EndPoint é a localización dende a que as API poden acceder aos recursos que necesitan para levar a cabo a súa función. 32, 33, 42, 43, 53

framework No desenvolvemento de software, un framework é unha estrutura conceptual e tecnolóxica de asistencia definida, normalmente, con artefactos ou módulos concretos de software, que pode servir de base para a organización e desenvolvemento de software. Tipicamente, pode incluír soporte de programas, bibliotecas, e unha linguaxe interpretada, entre outras ferramentas, para así axudar a desenvolver e unir os diferentes compoñentes dun proxecto. 16, 20, 50

FrontEnd O FrontEnd é a parte dunha aplicación que interactúa cos usuarios. É coñecida como "o lado do cliente". Basicamente é todo o que vemos na pantalla cando accedemos a un sitio web ou aplicación. 30, 33, 39, 40, 50, 82

Gateway Un Gateway é unha porta de enlace a outra rede. Funciona en todas as capas do modelo OSI e permite transferir paquetes a través de redes que empreguen un protocolo completamente diferente. A diferencia do Brdige, cámbiase o formato do paquete. 38, 39, 47, 82

Google Maps Google Maps é un servidor de aplicacións de mapas na web que pertenece a Alphabet Inc. 1, 3, 30, 33, 34, 61

Python Python é un lenguaxe de programación interpretado cuxa filosofía fai hincapé na lexibilidade de seu código. 13, 16, 40

Raspberry As Raspberry son unha serie de ordenadores de placa reducida, ordenadores de placa única e ordenadores de placa simple de baixo custo. 1–3, 8, 16, 33, 37, 47, 48, 65, 71

RFID Radio Frequency Identification é un sistema de almacenamento e recuperación de datos remotos que usa dispositivos denominados etiquetas, tarxetas ou transpondedores RFID. 1, 5, 7

ZigBee Zigbee é o nome da especificación de un conxunto de protocolos de alto nivel de comunicación sen fíos para su utilización con radiodifusión digital de baixo consumo, baseada no estándar IEEE 802.15.4 de redes sen fíos de área personal. 6, 7

Bibliografía

[1] J. Macias, "Bluetooth ble: el conocido desconocido," 2017, consultado o01/09/2020. [En liña]. Dispoñible en: https://solidgeargroup.com/bluetooth-ble-el-conocido-desconocido/

[2] GlobalTag, "Bluetooth ble: el conocido desconocido," 2019, consultado o02/09/2020. [En liña]. Dispoñible en: https://global-tag.com/es/tecnologia-ble/

[3] H.-S. C. for Astrophysics, "El gps y la relatividad," 2015, consultado o 03/09/2020. [En liña]. Dispoñible en: https://www.cfa.harvard.edu/space_geodesy/ATLAS/gps_es.html

[4] Comunidad, "Rfid," 2021, consultado o 03/09/2021. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/RFID

[5] E. de Expertos de la Universidad Internacional de Valencia, "Rfid: qué es y cómo funciona," 2017, consultado o 03/09/2021. [En liña]. Dispoñible en: https://www.universidadviu.com/es/actualidad/nuestros-expertos/rfid-que-es-y-como-funciona

[6] U. d. D. J. Fraile, "Zigbee vs bluetooth low energy," 2013, consultado o 23 de febreiro de 2021. [En liña]. Dispoñible en: https://blogs.deusto.es/aplicaciones-tic/zigbee-vs-bluetooth-low-energy/

[7] K. Shaw, "802.11: estándares de wi-fi y velocidades," 2018, consultado o 28/1/2021. [En liña]. Dispoñible en: https://www.networkworld.es/wifi/80211-estandares-de-wifi-y-velocidades

[8] "Mi maps: Aplicación basada en el envío de direcciones de google maps," consultado o 1/1/2021. [En liña]. Dispoñible en: https://www.researchgate.net/figure/Comparison-of-ZigBee-WiFi-and-Bluetooth_tbl10_287596223

[9] "Mi maps: Aplicación basada en el envío de direcciones de google maps," consultado o 18/2/2021. [En liña]. Dispoñible en: https://riunet.upv.es/bitstream/handle/10251/124626/Peris

[10] J. P. A. y. M. A. P. B. Vicente Cantón Paterna1, Anna Calveras, *A Bluetooth Low Energy Indoor Positioning System with Channel Diversity, Weighted Trilateration and Kalman Filtering*. Sensors MDPI, 2017. [En liña]. Dispoñible en: https://www.mdpi.com/1424-8220/17/12/2927

[11] *Localización en tiempo real de balizas a partir de rastreadores BLE usando Raspberry Pi*, consultado o 09/10/2020.

[12] "Atria," consultado o 28/1/2021. [En liña]. Dispoñible en: https://www.atriainnovation.com/sistemas-localizacion-interiores/

[13] "Situm," consultado o 28/1/2021. [En liña]. Dispoñible en: https://situm.es/es/sobre-situm

[14] "Bluerange," consultado o 28/1/2021. [En liña]. Dispoñible en: https://www.bluerange.io/asset-tracking/

[15] "Xeneral," consultado o 30/1/2021. [En liña]. Dispoñible en: https://www.bluetooth.com/bluetooth-resources/

[16] "Introduction to bluetooth low energy," consultado o 30/1/2021. [En liña]. Dispoñible en: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap

[17] "Rssi," consultado o 1/2/2021. [En liña]. Dispoñible en: https://www.speedcheck.org/es/wiki/rssi/

[18] "How to calculate distance from the rssi value of the ble beacon," consultado o 10/2/2021. [En liña]. Dispoñible en: https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/

[19] "Formula to convert the rssi value of the ble (bluetooth low energy) beacons to meters/feet," consultado o 10/2/2021. [En liña]. Dispoñible en: https://dzone.com/articles/formula-to-convert-the-rssi-value-of-the-ble-bluet

[20] "Signal strength and the rssi pin," consultado o 14/1/2021. [En liña]. Dispoñible en: https://www.digi.com/resources/documentation/Digidocs/90001456-13/concepts/c_rssi_pin_and_signal_strength.htm

[21] "Localization," consultado o 2/2/2021. [En liña]. Dispoñible en: https://github.com/kamalshadi/Localization

[22] "Trilateración," consultado o 28/1/2021. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/Trilateración

[23] "Multilateración," consultado o 28/1/2021. [En liña]. Dispoñible en: https://www.researchgate.net/figure/Figura-210-Metodo-Multilateracion_fig2_296195731

[24] "¿qué es una api de rest?" consultado o 3/2/2021. [En liña]. Dispoñible en: https://www.redhat.com/es/topics/api/what-is-a-rest-api

[25] "Meet django," consultado o 3/2/2021. [En liña]. Dispoñible en: https://www.djangoproject.com/

[26] "Sqlite," consultado o 3/2/2021. [En liña]. Dispoñible en: https://www.sqlite.org/index.html

[27] "Comparativa y análisis: Raspberry pi vs competencia," consultado o 24/01/2021. [En liña]. Dispoñible en: https://www.comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/

[28] "Raspberry s.o." consultado o 3/2/2021. [En liña]. Dispoñible en: https://www.raspberrypi.org/software/operating-systems/

[29] "Bluepy," consultado o 3/2/2021. [En liña]. Dispoñible en: https://github.com/IanHarvey/bluepy

[30] "ibks 105," consultado o 3/2/2021. [En liña]. Dispoñible en: https://accent-systems.com/es/producto/ibks-105/

[31] "Ibks105 datasheet," consultado o 24/10/2020. [En liña]. Dispoñible en: https://accent-systems.com/wp-content/uploads/iBKS105_datasheet_rev3.pdf

[32] "Batctl," consultado o 3/2/2021. [En liña]. Dispoñible en: https://github.com/open-mesh-mirror/batctl

[33] "B.a.t.m.a.n." consultado o 3/2/2021. [En liña]. Dispoñible en: https://www.open-mesh.org/projects/batman-adv/wiki

[34] "Vue.js," consultado o 3/2/2021. [En liña]. Dispoñible en: https://es.vuejs.org/v2/guide/#

[35] "Mqtt," consultado o 2/2/2021. [En liña]. Dispoñible en: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[36] "Introducción a los marcos Ágiles: Scrum," consultado o 24/10/2020. [En liña]. Dispoñible en: https://proagilist.es/blog/posts/introduccion-a-los-marcos-agiles-scrum/

[37] "¿qué eso de scrum?" consultado o 20/10/2020. [En liña]. Dispoñible en: https://richardgracia.com/scrum-para-startups/

[38] "Base de datos distribuida. ¿qué es? características," consultado o 10/2/2021. [En liña]. Dispoñible en: https://ayudaleyprotecciondatos.es/bases-de-datos/distribuida/#Ventajas_y_desventajas

[39] "Fundamento de las bases de datos: Modelo entidad-relación," consultado o 10/2/2021. [En liña]. Dispoñible en: https://www.genbeta.com/desarrollo/fundamento-de-las-bases-de-datos-modelo-entidad-relacion

[40] "Mtv y django," consultado o 20/10/2020. [En liña]. Dispoñible en: http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/

[41] S. Villodas, "Código fonte do tfg, localización mellorada mediante dispositivos ble," consultado o 23/02/2021. [En liña]. Dispoñible en: https://udcgal-my.sharepoint.com/:f:/g/personal/sergio_villodas_udc_es/EgOIQDfRnR5Ngbzm2vcgbhkB1P_KA-X8QZkmqo7cotGo_g?e=ARBaKf